Supplemento a ioProgrammo N°30

MAGAZINE

Novembre 1999

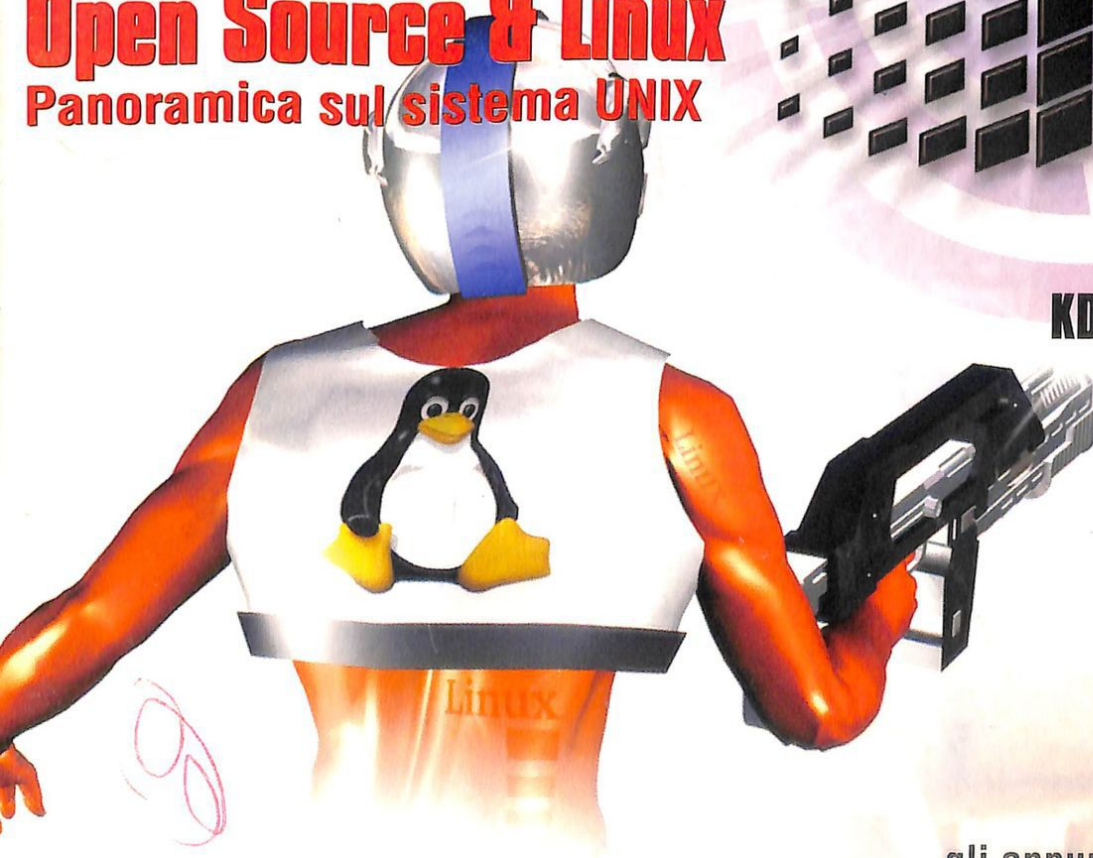
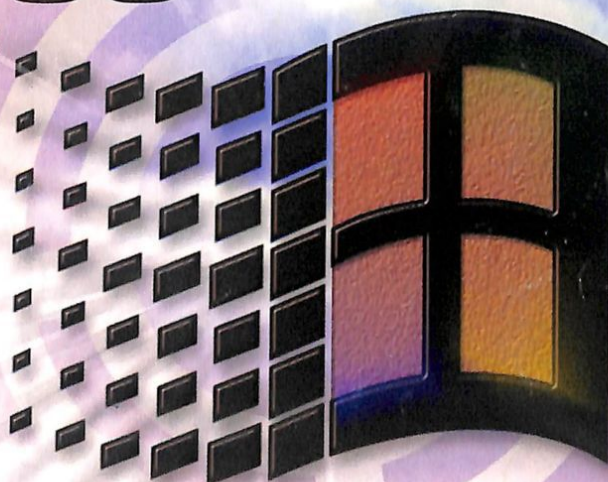
Linux Base

Come funziona e com'é strutturato

Free Software

La controversia COREL

Open Source & Linux

Panoramica sul sistema UNIX

Development

KDE: sviluppare applicazioni con l'AppWizard

PHP3: Web Scripting e Database

Administration

RPM: uno strumento potente e flessibile

Picnic:

gli appuntamenti del mondo Linux

Il Desktop ad una dimensione

"Una confortevole, levigata, ragionevole, democratica non-libertà prevale nella civiltà informatica avanzata, segno di progresso tecnico: Microsoft. Le accuse dell'anti-trust americano raccontano una storia; le note politiche commerciali del gruppo di Redmond, ne specificano gli aneddoti; il lock-in globale imposto agli utenti dai software di Zio Bill, ne sono solo un particolare. Non c'è nulla di male. La struttura economica di un mercato a rendimenti crescenti crea monopoli de facto, e gli utenti delle tecnologie dell'informazione non hanno esitato a consegnarlo, per i sistemi operativi desktop, a Microsoft, che ha dalla sua ha grandi software, prodotti concreti, una struttura aziendale di salute invidiabile, una mentalità positiva e ancora, all'apice del proprio successo, enormi potenzialità, ma soprattutto una solida realtà industriale. Ma Microsoft non piace. Non sono le dimensioni spropositate dei programmi tutto sommato banali che impone, né la cura maniacale dell'inutilità grafica delle proprie realizzazioni, e neppure la contorsione mentale a cui obbliga per configurare un altrimenti ovvio device millantato plug-and-play. Si sopportano pure i costi elevatissimi per gli upgrade che si succedono con precise cadenze biennali (e altrettanto precisi ritardi triennali) i cui budget pubblicitari superano di gran lunga quelli di sviluppo. Quello che non piace di Microsoft non è Microsoft in sé, ma il modello economico di cui oggi è la più grande sostenitrice al mondo. Open Source & Linux rappresentano l'alternativa. Dall'odierno 5%, Linux raggiungerà in fretta il 15% del mercato desktop, non tanto per il proselitismo millenarista dei pochi santoni e dei mille gregari, quanto per l'insofferenza d'alcune grandi aziende a cui Gates non sta proprio rendendo la vita facile e per l'inspiegabile euforia borsistica sui tecnologici che porta RedHat ad avere una capitalizzazione maggiore di SCO in tempi incredibilmente brevi (a quando l'OPS?) e, perché no, anche grazie a chi questo mondo s'impegna a spiegarlo "dal basso" come noi: programma per programma, linea di codice per linea di codice. È con le funzioni di un kernel, tra le limature di un pattern o nelle configurazioni di un request broker che oggi i non-violenti fanno le rivoluzioni. Non è improbabile che, con indicibile equilibrismo, stiamo solo cercando una via d'uscita da questa blockierte Gesellschaft (società bloccata) che abbiamo accettato sul piatto d'argento di M$."

Emmanuele Somma

Sommario

Anno I, N°2 Novembre 1999

Linux News

Cover story

ioProgrammo "Open Source & Linux"

Tecnica-Development

Tecnica-Administration

Spazio Picnic

Linux dati

Context

Supplemento ad ioPROGRAMMO N° 30

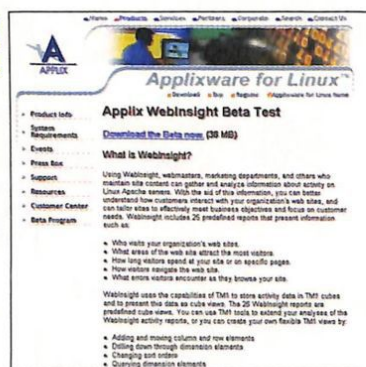
Direttore Editoriale
Massimo Sesti
Direttore Responsabile
Romina Sesti
Direttore Commerciale
Francesco Schirinzi
Responsabile di Produzione
Nicolino Rocca
Progetto e coord. grafico
Paolo Cristiano
Impaginazione elettronica
F. Ciliberti, L. Cocerio, A. Monaco
Coordinamento Editoriale
Fabio Farnesi
Coordinamento Redazionale
Emmanuele Somma
Redazione
F. Almagno, G. Forlino, T. Zaffino
Collaboratori
S. Frangella, M. Gastreghini, F. Marchetti Stasi, F. Munaretto, B. Parrella
Segreteria
Sandra Ionata
Redazione ioProgrammo
P.zza Libertà 35 - 87030 Rende (CS)
Tel. 0984/467948 r.a. - Fax 0984/467819
Posta elettronica: ioprogrammo@edmaster.it
Url: www.edmaster.it/ioprogrammo
Concessionaria esclusiva per la pubblicità
HOGA ITALIA S.p.A.
Piazza San Camillo De Lellis, 1
20124 - Milano - Tel. 02/66988424-5-6-7
Editore
Edizioni Master S.r.l.
Stampa Rivista
Roto Effe S.r.l. - Roma
Distribuzione per l'Italia
Parrini & C S.r.l. - Roma

Finito di stampare nel mese di Ottobre 1999

WebInsight disponibile in beta test

Applix, Inc. (NASDAQ:APLX), leader nel software applicativo per Linux, annuncia la libera disponibilità in Beta test di WebInsight

Con WebInsight è possibile valutare l'efficacia di siti internet o intranet anche dal punto di vista economico, poiché permette di determinare il comportamento degli utenti all'interno del sito e la loro provenienza, nonché il tempo di utilizzo dei servizi. I risultati sono rappresentabili sia in forma numerica che grafica. Peculiarità di WebInsight è la potenza delle analisi ad-hoc sull'uso del web, attraverso il motore analitico multidimensionale in-linea OLAP denominato iTM1 adottato da Applix. Per permettere una più ricca rappresentazione dei dati, WebInsight mostra i suoi dati in un completo spreadsheet di lavoro.

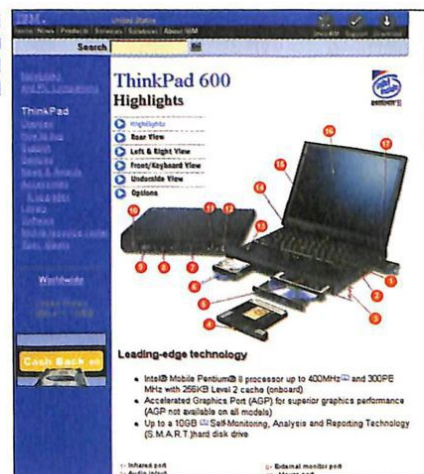
Pagina di WebInsight all'interno del sito ufficiale di Applix

WebInsight lavora su Apache per Linux sebbene in futuro sarà disponibile anche per altri sistemi operativi. La versione di beta test è disponibile per il download gratuito da *http://www.smartbeak. com/*. Nello stesso sito i beta tester avranno supporto o potranno riportare i loro suggerimenti e problemi. Dopo il periodo di beta test, presumibilmente a Dicembre, WebInsight sarà disponibile sui canali commerciali al prezzo orientativo di $695 per utente.

Per informazioni: http://www.applix.com

Primo LapTop compatibile con Red Hat Linux

Il ThinkPad 600 è il primo laptop ad essere certificato compatibile con Red Hat Linux

La certificazione dei laptop è sempre stata ostacolata dalla necessità di driver specifici. La mossa di IBM darà certamente una ulteriore mano a Linux nella conquista di posizioni di mercato tra gli utenti dei laptop oltre che dei desktop.

Overview di ThinkPad 600 all'interno del sito dell'IBM

IBM, peraltro, anticipa anche Dell che aveva annunciato la compatibiltà dei propri laptop della linea Inspirion, ma solo per la fine dell'anno.

Per informazioni: http://www.ibm.com

Linux, sistema preferito dagli sviluppatori

Secondo un recente studio Borland, sempre più sviluppatori considerano

Linux come sistema operativo per future applicazioni aziendali

Durante la Linux World Conference & Expo svoltasi in agosto a San Jose', California, Inprise Corporation (Nasdaq: INPR) ha ribadito il proprio impegno a supportare la piattaforma Linux. In tale occasione, l'azienda ha annunciato la disponibilita' immediata di Inprise VisiBroker per Linux, la nuova versione del celebre strumento object request broker (ORB), ed ha presentato in anteprima nella sezione showroom di Linux World il suo prossimo tool di sviluppo in Java, Borland JBuilder per Linux.

Sito italiano di Borland Inprise, la divisione Borland per l'impresa

Inprise Corporation ha inoltre reso pubblici i risultati completi di uno studio condotto su oltre 24.000 sviluppatori Linux, da cui emerge che Linux e' un sistema operativo cruciale per il futuro dei loro clienti. La maggioranza degli sviluppatori presi in analisi dallo studio sta pianificando su Linux lo sviluppo di applicazioni e di progetti database client server. Secondo un recente rapporto pubblicato da IDC (International Data Corporation http://www.idl.com), le vendite di un sistema operativo sono trainate dalle applicazioni e la disponibilita' di un numero maggiore di applicazioni su Linux fara' pertanto da traino a Linux nel settore del computing aziendale. ''InterBase per Linux e' gia' disponibile da oltre un anno. Con l'annuncio di VisiBroker per Linux, i nostri clienti possono ora accedere alla versione Linux di uno dei maggiori CORBA ORB per la distribuzione e l'installazione di applicazioni su piattaforme diverse'', ha osservato Dale Fuller, Presidente e CEO di Inprise Corporation. ''Sulla base dello straordinario feedback ricavato dal nostro studio, stiamo inoltre valutando quali altri tool e tecnologie Inprise puo' estendere a questo mercato in espansione''.
Secondo Ernesto Franchini di ISS (Agenzia esclusiva per l'Italia di Inprise Corporation) l'impegno di Inprise per Linux e' strategico anche per il mercato italiano, dato che ''Linux sta perdendo la connotazione di sistema operativo per sette segrete, e sempre piu' si identifica come una piattaforma con la P maiuscola che mette a disposizione prodotti e servizi''.

Per informazioni: http://www.inprise.it/

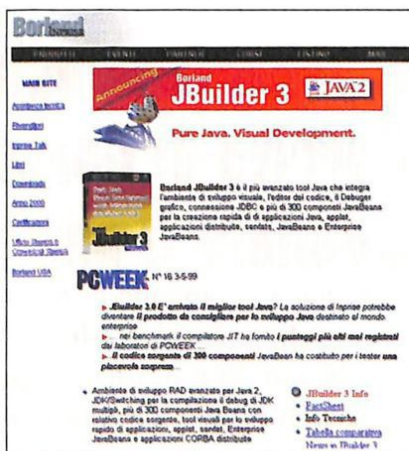
Inprise annuncia l'arrivo di Borland JBuilder 3 per Linux

Borland, la divisione di Inprise Corporation (Nasdaq:INPR), ha annunciato la prossima disponibilità di JBuilder 3 per sistemi Linux

La disponibilità per Linux di VisiBroker è il primo passo verso una strategia globale di Inprise verso la Piattaforma Linux che perterà presto al rilascio di Borland JBuilder 3: la nuova versione del tool di sviluppo visuale per la creazione di applicazioni database ed enterprise Java indipendenti da piattaforme. Borland JBuilder 3 fornisce completo supporto per la piattaforma Java 2 e permette agli sviluppatori singoli e aziendali di creare piu' facilmente e rapidamente applicazioni database ed enterprise indipendenti da piattaforme, applicazioni enterprise distribuite e componenti JavaBean.
JBuilder 3 e' stato progettato per essere disponibile su piattaforme multiple: dapprima su Microsoft Windows, poi su Solaris

entro la fine dell'anno, ed infine su Linux.
Borland JBuilder 3 comprende:

- supporto completo per la piattaforma Java 2 per creare applicazioni Java affidabili e scalabili scritte interamente in linguaggio Java;
- tool visuali e componenti riutilizzabili per la creazione rapida di applicazioni indipendenti da piattaforme, servlet e applet;
- supporto CORBA integrato e automatizzato per ridurre notevolmente tempo e sforzi richiesti per sviluppare e implementare client, server e servlet CORBA;
- Wizards e Visual Designers per la creazione di JavaBeans ed Enterprise JavaBeans riutilizzabili.

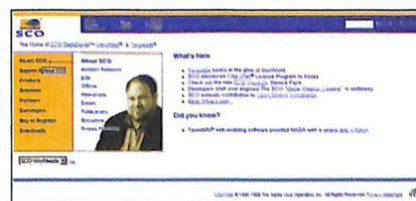
Pagina di Borland JBuilder 3, all'interno del sito inprise.it

L'ambiente aperto di JBuilder 3 supporta anche JDK 1.1.x, componenti JFC/Swing, JavaBeans, Enterprise JavaBeans, CORBA, RMI, JDBC e tutti i principali server di database aziendali. JBuilder 3 sara' proposto nelle tre versioni Enterprise, Professional e Standard ai rispettivi prezzi di Lire 4.990.000, Lire 1.460.000 e Lire 199.000.
In Italia, Inprise opera attraverso l'agenzia esclusiva ISS (International Software Services), che ne cura le attivita' promozionali, commerciali e di marketing, coordinando i partner commerciali e tecnologici della societa'.

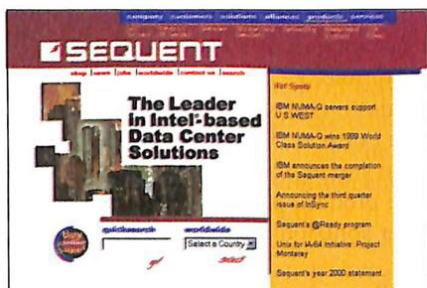
Per informazioni: http://www.inprise.it/

In arrivo Unix/64 per Intel Merced

Monterey/64: il primo sistema operativo UNIX a girare su Merced di Intel

Monterey/64 è il primo sistema operativo UNIX a girare su Merced di Intel. Lo hanno annunciato IBM, Intel, SCO e Sequent dopo che il boot del sistema operativo sul chip di Intel e' stato completato presso i laboratori di Intel a Dupont, Washington.
Questo risultato sottolinea l'impegno dei partner del Progetto Monterey nel rendere Monterey/64 il sistema operativo UNIX high-volume di classe enterprise per le architetture IA-64 di Intel e i processori Power di IBM.
In occasione del Developer's Forum di Intel, svoltosi il mese scorso a Palm Springs, in California, IBM, SCO e Sequent hanno effettuato una dimostrazione di Apache Webserver su Monterey/64 sul simulatore di Merced. Il porting di Apache Webserver a Monterey/64 e' stato completato in meno di un giorno, a garanzia di un'immediata migrazione delle applicazioni a Monterey/64.

Sito ufficiale di SCO, azienda che partecipa al progetto Monterey

Il successo del Progetto Monterey continua a crescere a pieno ritmo, come dimostrano i numerosi fornitori di hardware che hanno aderito al progetto: Acer, Bull, Compaq, CETIA (sussidiaria di Thomson-CF), IBM Netfinity servers, ICL, Sequent, Unisys e, piu' recentemente, Samsung Electronics. Numerosi, inoltre, gli sviluppatori e fornitori indipendenti che continuano ad aggiungersi ai partner del progetto.
Cygnus Solutions, societa' leader nel software Open Source, ha recentemente

annunciato un accordo per offrire i tool di sviluppo Cygnus GNUPro ottimizzati per il sistema operativo AIX su architetture Power e IA-64. Edinburgh Portable Compilers (EPC) ha inoltre annunciato il suo supporto al Progetto Monterey. EPC ha reso note le proprie stategie per rendere disponibili le versioni native e cross-compiling del suo kit di tool C/C++ e Fortran per la linea di prodotti Monterey.

Sito ufficiale di Sequent, altra azienda partecipante al progetto Monterey

Annunciato lo scorso ottobre, il Progetto Monterey e' un'iniziativa congiunta di IBM, SCO, Sequent e Intel per lo sviluppo di un sistema operativo UNIX high-volume di fascia enterprise in grado di girare sui processori IA-32 e IA-64 e i processori Power di IBM, in sistemi che vanno dai server dipartimentali a quelli dei centri dati. Questa iniziativa prevede inoltre lo sviluppo di un sistema operativo UNIX in volumi per processori Intel IA-64, utilizzando IBM AIX, SCO UnixWare e le tecnologie enterprise di Sequent.
I principali fornitori di software che hanno gia' aderito al Progetto Monterey comprendono: Baan, BEA Systems, Compuware, Cygnus Solutions, Data Pro Accounting Software, Informix, Merant Micro Focus, Netscape Communications, Novell, PeopleSoft, Pick Systems, Progress Software, Rational, Real World, Risk Management Technology, Software AG, SAS Institute, Take Five, Tivoli, Viador, BMC Software, daly.commerce, Facet, Geac SmartEnterprise Solutions, ISOCOR, J.D.Edwards, Marcam Solutions, Parasoft, Sanchez, Sapiens International, Sendmail e TIBCO.

Per informazioni: http://www.sequent.com
http://www.sco.com

Il prezzo del software crollerà. È solo una questione di tempo

"Ci sono segni inequivocabili. Entro tre anni sarà tutto diverso." dice Linus in una intervista alla Reuters

Dopo che l'Università di Stoccolma gli ha conferito il dottorato onorario per la creazione di Linux, Linus, in una intervista alla Reuters, parla della crescita della domanda di software basato sulle richieste dell'utente in particolare per gli embedded system. "Il sistema di pricing del software verrà sostanzialmente modificato dall'ampia disponibilità di software libero da specializzare". L'attenzione della comunità dell'Open Source verso il mercato di tool specializzati renderà semplice ad un sistema operativo come Linux di passare dai server, dove oggi ha acquisito una importante parte del mercato, a quella dei desktop dove il dominio di Windows sarà messo in seria difficoltà.
"Tra tre anni vedremo..." è convinto Linux, sebbene il centro dei suoi pensieri siano comunque i sistemi custom. È una scelta di campo netta per un futuro digitale dove la presenza di molti dispositivi diversi e separati giocherà un ruolo essenziale.
L'integrazione oggi possibile non porterà, secondo il creatore di Linus, alla richiesta di strumenti unificati per tutte le situazioni. "Per esempio il Nokia 9000 (Communicator) è interessante e mi piace,

Sito di Reuters, l'azienda svedese leader nel campo della informazione

ma finisce per non essere né un buon cellulare né un organizer decente". Allo stesso modo critica gli sforzi per avere un unico apparecchio centralizzato familiare sia per la televisione digitale che per l'accesso ad Internet. Il futuro sarà di terminali separati che condividono l'informazione. E si chiede: "Altrimenti che succederà quando qualcuno in famiglia vorrà vedere la televisione e qualcun'altro cercare qualcosa in Internet?"

Per informazioni: http://www.reuters.com

Progress Software offre oltre 5.000 applicazioni commerciali per piattaforma Linux

Il principale fornitore di tecnologia per pacchetti applicativi apre la strada all'offerta di migliaia di applicazioni commerciali sotto Linux

Progress Software Corporation (NASDAQ: PRGS), fornitore leader di prodotti per lo sviluppo, il deployment e la gestione di applicazioni, ha annunciato che offrira' nuovi prodotti ai suoi Independent Software Vendor (ISV) e utenti finali per aiutarli a portare oltre 5.000 applicazioni commerciali sul sistema operativo Linux.

Tali prodotti includono: Progress(R) Open AppServerTM, un server applicativo per il deployment di componenti applicativi condivisi su ambienti eterogenei e il database scalabile Progress(R) RDBMSTM, per la gestione e la memorizzazione dei dati di applicazioni SQL e 4GL.

Progress Software Corporation e' un fornitore mondiale di prodotti e servizi software

Sito ufficiale di Progress Software, produttore di oltre 5.000 applicazioni per Linux

per lo sviluppo, il deploying e la gestione di applicazioni mission critical che funzionano su qualsiasi piattaforma informatica o ambiente di rete.

I suoi prodotti, che sono installati in oltre il 60% delle Fortune 100 company, includono application server, database, strumenti di sviluppo e prodotti per la gestione di applicazioni per soluzioni Internet/Web, extranet e intranet.

I prodotti Progress sono principalmente utilizzati da Fornitori di Software Indipendenti (ISV), che a loro volta generano ogni anno vendite di applicazioni per circa 2 miliardi di dollari. L'azienda, nata nel 1981, ha oltre 1.200 dipendenti e una presenza commerciale in piu' di 100 Paesi nel mondo.

Per informazioni: http://www.progress.com/

Schede PCI per Linux

FITWIN PCI di Impact Technologies disponibili per Linux

Le schede Fitwin PCI di Impact Technologies, nella versione a 4 e 8 porte seriali, disponibili dal maggio scorso in ambiente Microsoft (95, 98 e NT), Prologue e Unix S.C.O. (Open Server 5.0), sono oggi proposte sotto Linux.

In Italia sono distribuite da Prologue Italia. La gamma Fitwin PCI e' composta da 2 modelli: Fitwin PCI4 (4 porte seriali), Fitwin PCI8 (8 porte seriali). Elaborate a partire da un circuito specializzato ad alta performance, che permette un trattamento molto rapido dei dati, queste schede raggiungono velocita' di trasmissione fino a 230 Kbauds per ogni porta, riducendo nel contempo il carico di lavoro del processore

Pagina Web della scheda FitWin PCI nel sito ufficiale di Impact Technologies

dell'unita' centrale. Grazie all'elevata ergonomia del configuratore, l'installazione delle schede si realizza in poco tempo. Come tutte le schede di IMPACT TECHNOLOGIES, possiedono uscite protette contro le sovratensioni, che garantiscono un funzionamento perfettamente affidabile in ambiente perturbato.

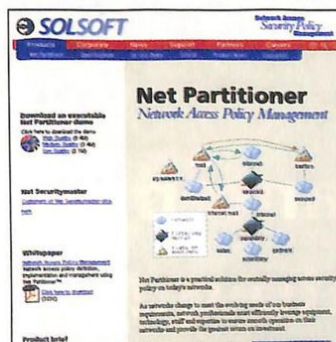
Per informazioni: http://www.itechno.com/

Network Security Managment di SolSoft: Net Partitioner

Solsoft lancia in Italia Net Partitioner, una collaudata e affidabile soluzione di network security management basata su criteri

Net Partitioner offre ai professionisti una potente soluzione basata su criteri che consente di automatizzare le funzioni di security management a livello di network, pur assicurando la flessibilita' richiesta dalle mutevoli esigenze aziendali.
Fornendo la capacita' di gestire i criteri per la protezione dell'accesso centralizzato su reti eterogenee, Net Partitioner riduce fino al 90% i tempi di implementazione e gestione. Tradizionalmente, ci si aspetta che i firewall forniscano la protezione necessaria ad evitare intrusioni dall'esterno nei confini del network.
Le attuali statistiche evidenziano tuttavia una realta' secondo cui oltre la meta' delle violazioni delle informazioni protette nascono all'interno dei confini del network e non vengono rilevate e controllate dai firewall.
Sempre piu' spesso nei network aziendali i vantaggi derivati dal consentire ai partner l'accesso attraverso i confini del network devono essere bilanciati da una metodologia di protezione in grado di respingere azioni potenzialmente dannose per le zone protette del network.
Net Partitioner e' la prima soluzione di enterprise network security basata su criteri, vale a dire il partizionamento del network in zone rese sicure mediante l'implementazione capillare di criteri di sicurezza basati su filtri IP o ACL (Access Control List). Le principali funzionalità del prodotto sono:

- Gestione centralizzata e globale dei criteri;
- Definizione grafica dei criteri ;
- Automazione di criteri ''network aware'';
- Management e controlli;
- Supporto per reti eterogenee.

Net Partitioner fornisce una visione globale

Pagina Web del prodotto, all'interno del sito di SolSoft

della rete basata su criteri e una console per il security management centralizzato dell'accesso.
Net Partitioner puo' essere eseguito sui piu' diffusi sistemi operativi, tra cui Windows 95/98, Windows NT, Sun Solaris, AIX, HP-UX, Linux e BSD.
Supporta tutti i protocolli IP personalizzati e standard, e' estremamente scalabile e non pone alcun limite al numero di dispositivi o alla dimensione del network che puo' supportare.

Per informazioni: http://www.solsoft.com

Un pinguino nel bazar

Linux non è semplicemente un sistema operativo: è una filosofia, l'incarnazione del free software, il prodotto della cultura hacker, il sogno di ogni programmatore divenuto realtà. Linux non è solo Linux, non è neanche solo Unix: nato dal connubio tra gli esperimenti di un giovanissimo programmatore finlandese, non potrebbe esistere senza il software sviluppato lungo un arco di quasi venti anni da un hacker che propugna la causa della libertà del software, e basa la sua interfaccia grafica su un sistema sviluppato al MIT lungo lo stesso arco di tempo. La sua storia comprende un arco di appena otto anni, e lo ha recentemente portato sulle prime pagine delle riviste di tutto il mondo; la sua disponibilità gratuita su un enorme numero di piattaforme lo ha reso sempre più popolare tra gli appassionati di informatica, e lo ha portato lentamente a spiazzare versioni di Unix apparentemente ben più antiche e blasonate, fino ad arrivare a sfidare – secondo quanto dicono i giornali – anche il mercato dei personal computer, dominio quasi incontrastato della Microsoft. La distanza che separa Windows da Linux è ancora estremamente ampia: sistema orientato alla facilità d'uso il primo, con un'interfaccia semplice ed apparentemente intuitiva che però nasconde la maggior parte dei dettagli di funzionamento; sistema ostico e difficile da imparare il secondo, ma di grande duttilità e con tutti i dettagli disponibili pubblicamente. Laddove sembra difficile che Windows possa essere modificato, in modo da consentirne una più profonda adattabilità, o che ne venga reso disponibile il codice sorgente, Linux sta lentamente percorrendo la strada per poter presentare all'utente finale un'interfaccia semplice, aggiungendo un nuovo strato di software grafico: l'ambiente di desktop Gnome. L'aspetto finale ricorda quello di tutti i sistemi operativi orientati alla grafica, con la differenza che i componenti sono ben visibili all'analisi, e che l'interfaccia può essere facilmente eliminata o sostituita. Il progetto Gnome è ancora in una fase di relativa immaturità ma, i risultati raggiunti sono già eccellenti, e costituiscono un primo passo importante verso un Linux di più semplice utilizzo. Allo stato attuale delle cose, comunque, possono esservi numerose situazioni nelle quali l'utilizzo di Windows è maggiormente consigliabile; ne parleremo nel primo articolo. Storicamente Unix è stato il sistema operativo dei laboratori di ricerca, dei centri di calcolo, della cultura hacker degli ultimi trent'anni; ed è stata proprio la cultura hacker che ne ha consentito la nascita, non appena i personal computer sono divenuti sufficientemente potenti da permettere l'utilizzo di software più sofisticato. Ma la cultura hacker ha dato un contributo ancora più importante alla nascita di Linux: da essa è nata, infatti, l'ideologia del free software, personificata nella figura quasi leggendaria di Richard Stallman. La Free Software Foundation, da lui fondata, ha perseguito l'obiettivo di un sistema Unix liberamente disponibile lungo un arco di tempo di quasi vent'anni, ed è stato solo grazie ad essa che Linux ha potuto vedere la luce e trovare la maggior parte delle componenti, necessarie a un sistema operativo, già pronte. Sono aspetti che approfondiremo nel secondo articolo di questo speciale, con particolare riferimento alla recente polemica suscitata dalla Corel, accusata di violazione di copyright… o dovrei piuttosto dire di copyleft! Ma quali sono stati gli eventi che hanno portato allo sviluppo di Linux? E quali gli elementi essenziali che lo hanno reso possibile? Nella maggior parte dei resoconti si dà solo un rapido accenno al ruolo svolto da Linus Torvalds e all'assistenza prestata dagli hacker attraverso Internet; viene raramente analizzato in che modo questo sia stato possibile. Tra le persone che si sono poste questa domanda c'è stato anche un programmatore di free software che ha paragonato il modello di sviluppo di Linux a un rumoroso bazar, per contrasto con il rigoroso approccio classico; delle sue idee e dell'influenza che hanno avuto sul mondo dell'Open Source, parleremo nell'ultimo articolo.

Francesco Marchetti Stasi

... e Linux regna!

Il confronto tra Windows e Linux provoca spesso aspri conflitti tra i sostenitori dei due sistemi; in realtà, al momento attuale ciascuno di essi trova la sua ragione d'essere in contesti diversi. Ma a lungo termine Linux potrebbe essere il favorito...

Meglio Linux o Windows? Questa domanda ha ormai lasciato il ristretto circolo delle newsgroup di Internet per farsi strada fino ai piú popolari settimanali, dove i toni infuocati si stemperano in una sorta di mite stupore per il successo che un sistema operativo "gratuito" è riuscito ad ottenere.
Quando si torna all'ambiente tecnico, comunque, la domanda è in grado di suscitare dibattiti infiammati, in special modo tra i "paladini del software libero" e i "sostenitori delle interfacce intuitive"; personalmente, ritengo che la domanda "qual è migliore?" sia basilarmente mal formulata, in quanto ciascuno dei due sistemi è piú adatto in ambiti diversi. Inoltre, quando si considera la scelta non per sé ma per altre persone, magari un'azienda di una certa vastità, bisogna dimenticare le proprie personali preferenze e considerare pro e contro con la massima obiettività; e in fin dei conti non è difficile, con un po' di buon senso, ristabilire meriti e demeriti di ciascun sistema operativo e comprendere quale sia l'ambito piú adatto ad ognuno.

Linux contro Windows

Per sgombrare il campo da ogni equivoco, inizierò con il dichiarare la mia devozione di lunga data (sulla scala dei tempi informatici...) al Software Libero in generale e a Linux in particolare; il mio utilizzo di Windows, invece, è puramente professionale e avviene senza la minima gioia, ma è sufficiente a farmi apprezzare la sua utilità in certi ambiti. Il caso piú immediato che viene in mente è quello di una qualsiasi società che abbia le piú elementari esigenze di ufficio: produrre documenti, tabulati, relazioni, presentazioni, e cosí via. È vero che esistono pacchetti per l'automazione d'ufficio anche per Linux, ma la grande diffusione di Microsoft Office lo rende praticamente una scelta obbligata per le aziende che abbiano necessità di scambiare documenti con l'esterno: cioè praticamente tutte. Si assiste qui a un fenomeno curioso per cui l'interoperabilità viene ottenuta non attraverso l'utilizzo di formati standard per i documenti, come sarebbe lecito aspettarsi, ma

piuttosto attraverso il monopolio di una singola azienda, dei suoi formati privati e dei suoi prodotti. Questa situazione è comunque in evoluzione, per quanto lenta e non evidentissima. Un primo passo è stato compiuto dall'introduzione, da parte della Adobe, del formato pdf, le cui specifiche sono state rese pubbliche; esso consente la descrizione delle pagine di un documento esattamente nel formato in cui l'autore le ha create, ma a differenza del Postscript, pure ideato dalla Adobe, consente un'efficiente visualizzazione a schermo ed è adatto all'utilizzo su Internet. L'utilizzo del pdf come formato per lo scambio di documenti che non è necessario modificare è un passo importante verso l'interoperabilità. Un altro evento di particolare importanza è il fatto che Microsoft Office 2000 usa come formato standard per i documenti l'XML; è quindi prevedibile che in un tempo non troppo lungo verranno create applicazioni per Linux in grado di creare o leggere documenti Office. Non tratterrei il fiato nell'attesa, ma sono moderatamente ottimista.

Figura 1 – Il sito http://srom.zgp.org/ utilizza il motore di ricerca Altavista per ricercare le pagine in cui si dice che un sistema operativo "fa schifo" ("sucks") o "è grande" ("rocks", "rules"). Windows e Linux sono quelli sui quali si discute maggiormente, e sembra che chi parli bene dell'uno parli male dell'altro...

Quanto detto sopra non vuol dire che per tutte le aziende è obbligatorio l'utilizzo di Windows!

Possono esserci motivi ideologici per cui un'azienda decide di utilizzare esclusivamente software libero; negli ambienti della ricerca scientifica, ad esempio, gli utenti sono legati ai sistemi operativi di tipo Unix, e hanno visto l'avvento di Linux come l'occasione per poter finalmente avere questo tipo di sistema operativo su stazioni economiche e potenti; infine, se un'azienda ha necessità molto specifiche, per le quali è comunque necessario lo sviluppo di software "ad hoc", e come politica generale non desidera che sulle postazioni di lavoro venga installato software non autorizzato, può ritenere opportuno l'utilizzo di Linux. In linea di massima, comunque, questa scelta è ancora estremamente rara, o circoscritta ad ambienti estremamente specifici. Un'altra considerazione che viene spesso fatta, nella scelta di un ambiente Windows, è la maggiore reperibilità di assistenza specializzata; questo è un tipo di considerazione che io personalmente non ritengo di gran rilievo, perché se è apparentemente piú facile trovare amministratori per Windows NT che per Linux, è anche vero che il livello medio di preparazione è decisamente piú elevato tra i secondi; inoltre, la disponibilità di assistenza per un Sistema Operativo dipende dal mercato, e in effetti negli ultimi anni stiamo assistendo a una rapida crescita delle aziende che offrono supporto per Linux.

Anche per gli utilizzi non professionali viene sempre o quasi sempre installato Windows, e il motivo è ben chiaro: il venditore preferisce un sistema che gli semplifichi l'installazione e l'assistenza; l'utente è in genere interessato alle applicazioni, e non ha alcun motivo di preferire un sistema operativo ad un altro. Il sistema non contiene un programma di fotoritocco? Un amico o un venditore ambulante sarà pronto a fornirgli una copia gratuita o molto economica di qualche celebre applicativo; con l'accesso a Internet è possibile procurarsi, con grande facilità, applicazioni meno complesse, oppure versioni dimostrative corredate dall'apposito crack. Naturalmente, si finisce con il muoversi ai margini della legalità, ma di una legalità troppo sfumata per apparire come un vero crimine, e che appare confezionata per compiacere i produttori di software più che per fare

Domain : RIPE+.edu
DName : ---------
Service : ftp+news+www
Date : April '99

Host OS recognized (grouped, sorted)		01/99	%recog	04/99	%recog	+/-%
1. ( )	Linux	287093	28.5	399748	31.3	+2.8
2. ( )	Windows 95/98/NT	245184	24.4	310162	24.3	-0.1
3. ( )	Solaris/SunOS	178350	17.7	213304	16.7	-1.0
4. ( )	BSD Family	150961	15.0	186385	14.6	-0.4
5. ( )	IRIX	52941	5.3	58112	4.6	-0.7
6. (7.)	Mac/Apple	16170	1.6	26548	2.1	+0.5
7. (6.)	AIX	17362	1.7	18427	1.4	-0.3
8. (9.)	HPUX	10416	1.0	11760	0.9	-0.1
9. (8.)	Reliant Unix/Sinix	12848	1.3	10616	0.8	-0.5
10. (12.)	Novell Netware	3656	0.4	5982	0.5	+0.1
11. (10.)	Digital Unix	5016	0.5	5186	0.4	-0.1
12. (11.)	SCO Unix	3787	0.4	4021	0.3	-0.1
	++checksums	983784	97.8	1250250	97.9	

Figura 2 – Il sito http://leb.net/ utilizza il programma "queso" per stabilire quale sistema operativo venga utilizzato sui server di Internet. Linux domina la classifica, ed è in costante crescita relativa; Windows nelle sue varie forme è buon secondo. Si noti anche il buon posizionamento della famiglia BSD.

appello ad obiettivi principi di giustizia.

Linux e l'amministratore di sistema

Solo il più indifferente degli utenti lavorerà più di sei mesi con Windows senza imparare almeno un minimo di principi amministrativi: un'organizzazione razionale delle cartelle, come installare e disinstallare applicativi, come configurare le periferiche; i più smaliziati potranno magari smanettare anche con il registro. Queste esperienze possono essere interessanti ma mostrano ben presto i loro limiti: il computer a volte reagisce nella maniera appropriata ma, altre volte si comporta in modo apparentemente casuale, e non si riesce a capirne il perché. Tutta la complessità è stata così ben nascosta sotto la superficie levigata dell'interfaccia grafica che ricostruire il comportamento del computer è praticamente impossibile; e inizia a prendere corpo l'impressione che non ci siano regole che tengano, salvo la completa casualità.

Nel mondo di Linux, al contrario, qualunque comportamento è sempre ricostruibile e spiegabile: la maggior parte delle applicazioni –quelle più vicine all'hardware– sono semplici, lineari e molto piccole. Il codice di tutte le applicazioni è disponibile, è possibile studiarlo, modificarlo, osservarne il comportamento con un debugger o intercettare le sue chiamate al sistema operativo, tecnica molto efficiente per capire rapidamente perché un'applicazione si inceppa. Notare che ho detto "un'applicazione": il sistema operativo non si blocca praticamente mai, almeno se si decide di non vivere pericolosamente e si utilizzano le versioni "stabili". Rispetto al mondo Windows, si può ben dire che Linux sia il paradiso del programmatore: è possibile studiare, modificare e riadattare il codice di qualunque parte del sistema, e il numero e la qualità degli strumenti di programmazione sono tali che a volta diventa difficile scegliere il più adatto.

Cenni storici...

Naturalmente, Linux non nasce come sistema operativo "didattico", a differenza del suo più diretto antenato. Quando nel 1990 Linus Torvalds comprò il suo 386, come migliaia di studenti intorno al mondo vi installò il sistema operativo Minix, scritto da Andrew Tanenbaum a scopo meramente didattico, a tutt'oggi ancora disponibile in commercio con lo stesso scopo. Fu proprio per superare i limiti intrinseci di Minix che Linus iniziò a sviluppare un nuovo sistema Unix, finché nel settembre del 1991 non fu in grado di distribuire la prima versione del sistema operativo destinato a dargli la celebrità. Immediatamente, decine di appassionati iniziarono a lavorarci a loro volta, inviando all'autore aggiunte, modifiche e correzioni, che egli inseriva immediatamente, o anche semplici "bug report", che venivano corretti a velocità impressionante. È rimasta celebre la sua risposta a uno di essi: Sono un idiota... ci ho messo circa cinque minuti per trovare l'errore.

La riprova della solidità tecnica di Linux sta nel numero di adattamenti per altre piattaforme che sono stati prodotti: ARM, Alpha Digital, Sparc SUN, M68000, MIPS e PowerPC. Con la maturità tecnica di Linux, ogni gruppo interessato a un sistema operativo libero, per una certa architettura,

aveva la possibilità di non partire da zero, e quindi un carico di lavoro molto minore per ottenere un sistema funzionante. A loro volta, questi adattamenti mettevano in luce gli aspetti di Linux piú legati all'architettura originale e i bachi più oscuri; il risultato finale è oggi, letteralmente, davanti agli occhi di tutti: un sistema operativo moderno, versatile, multiarchitetturale, stabile e affidabile.

Linux come server

Mentre come piattaforma base per le società Linux non è in genere consigliabile, e all'utente domestico poco interessato all'informatica potrebbe causare crisi isteriche, vi sono situazioni professionali in cui l'utilizzo di Linux è altamente caldeggiato. Un tipico esempio è quello dei server applicativi, in particolare dei server Web; quest'ultimo caso è uno dei piú eclatanti, dal momento che i sistemi Linux con il server Web Apache sono i più diffusi di Internet. L'utilizzo di Linux come file server è essenziale negli ambienti misti Windows/Unix, dal momento che Windows NT non è in grado di funzionare come file server per macchine Unix; l'installazione di un server di database su piattaforma Linux, già resa possibile da un certo numero di applicativi liberi, si è ampliata di recente con le versioni per Linux dei piú diffusi produttori di database. Altre applicazioni ancora più comuni includono l'utilizzo come server DHCP e DNS, come proxy Web, come router per l'accesso a Internet, eccetera...

Figura 1 – Il sito http://counter.li.org/ tenta di mantenere un database di utenti di Linux intorno al mondo – correte a registrarvi!

C'è un ultimo esempio di utilizzo di Linux che voglio discutere, perché mi riguarda particolarmente da vicino; si tratta dello studio e dell'osservazione di una rete, locale o geografica, sia dal punto di vista dell'efficienza che da quello della sicurezza. Al giorno d'oggi, Linux ha praticamente il monopolio degli strumenti di questo tipo: se viene individuato un baco nell'implementazione dei protocolli di rete su qualsiasi piattaforma, state certi che dopo ventiquattr'ore troverete un programmino compilabile sotto Linux che lo sfrutta. Volete osservare il traffico in transito su una rete? ci sono almeno tre strumenti di questo tipo per Linux. È possibile accorgersi se qualcuno sta spiando la vostra rete con la stessa tecnica? Ci sono due programmi per Linux che danno questa risposta. Senza contare, naturalmente, che potete creare un pacchetto dati da zero, non solo con un apposito programma, ma anche copiando il codice dal kernel, con il tradizionale ricorso alle system call, con le apposite librerie per Perl, e chi piú ne ha piú ne metta!

E per un programmatore, è più consigliabile operare in ambiente Windows o Linux? In questo caso sarò salomonico, rispondendo "tutt'e due". Linux consente di imparare a operare in ambiente Unix, sia dal punto di vista dell'utente che dello sviluppatore, ed è quindi un'esperienza professionale pressoché essenziale; inoltre, non va trascurato l'aspetto didattico di cui ho già parlato. L'ambiente di programmazione è più semplice e lineare, la scomposizione in componenti piú chiara, l'interfaccia con l'hardware più immediata: i sistemi di tipo Unix sono storicamente i più adeguati per studiare il C, ma utilizzando esclusivamente software libero è possibile imparare con implementazioni di alta qualità anche java, il C++, il lisp, il Fortran, l'Objective C, il Pascal. Per realizzare interfacce grafiche si utilizza X Windows, un sistema indipendente dall'hardware e dal sistema operativo che costituisce

lo standard "de facto" per tutti i produttori Unix. Le librerie disponibili sono ancora più numerose: praticamente tutti i programmi di una certa complessità sono accompagnati dalla corrispondente libreria. Le più importanti sono quelle utilizzate per la programmazione in C++ (STL, "Standard Template Library"), per lo sviluppo di applicazioni X Windows, per la gestione di immagini, per l'ambiente di Desktop Gnome, per le applicazioni Web e per la gestione di database. La creazione di "script" può essere effettuata con una shell di comandi, oppure con i piú potenti Perl, Python o Tcl, tutti dotati di estensioni grafiche per X Windows.

Conclusioni

Insomma, gli strumenti di programmazione per Linux sono numerosissimi, affidabili e potenti; l'amministrazione è indubbiamente più difficile che in Windows ma, offre anche possibilità illimitate; i server applicativi vengono sempre più frequentemente installati su piattaforma Linux; la documentazione è liberamente disponibile e di ottima qualità; cresce il numero di aziende che offrono assistenza ma, per le persone dotate di competenza tecnica, l'aiuto più valido è quello offerto dai newsgroup di Internet. Non vi resta che continuare a seguirci per avere una guida in questo vasto e variegato universo, e raccogliere anche dell'ottima documentazione in italiano!

Francesco Marchetti Stasi

I rivali di Linux

A parte il fatto che Linux ha un nome figo, c'è qualche motivo perché dovrei usare Linux piuttosto che BSD?

No. Questo è tutto. Il nome figo, voglio dire. Abbiamo lavorato duro per creare un nome che piacesse alla maggioranza della gente, e certamente ne è valsa la pena: migliaia di persone usano linux solo per poter dire "OS/2? Ha! Io uso Linux. Che nome figo". 386BSD ha fatto l'errore di mettere un mucchio di numeri e abbreviazioni strampalate nel nome, e spaventa un mucchio di gente perché suona troppo tecnico. (*Risposta di Linus Torvalds su un newsgroup di Internet*)

Tutto questo accadeva un po' di anni fa; da allora le cose sono cambiate in meglio per lo sfortunato 386BSD, che si è evoluto dando origine a due sistemi operativi dal nome decisamente piú pronunciabile: FreeBSD e NetBSD. Purtroppo, hanno qualche difficoltà a liberarsi dal BSD nel nome; anche da questo punto di vista la scelta di Linus di fare riferimento allo standard Posix, è stata decisiva...
Ricordate, quindi: quando dovrete scegliere un nome per l'applicazione che deve rendervi celebri (come Linus) o miliardari (come Bill Gates), scegliete un nome pronunciabile. C'è qualche buon motivo per scegliere FreeBSD o NetBSD anziché Linux? Non molti, decisamente: superficialmente tutti e tre i sistemi si somigliano molto, anche perché, kernel a parte, la maggior parte delle applicazioni è esattamente la stessa. L'unico punto in forte favore di Linux è la diffusione: pur non esistendo dati attendibili sul numero di utenti, basta guardarsi un po' intorno per rendersi conto che il numero di appassionati degli altri due sistemi è decisamente ridotto. Perciò, la scelta è abbastanza semplice: se volete un sistema per cui sia piú facile trovare driver e collettivamente ben noto, Linux è la vostra scelta, e una volta fatta una certa esperienza, nulla vi impedirà di valutare personalmente anche gli altri due sistemi; se invece desiderate un sistema operativo che ancora non ha conquistato la ribalta della cronaca, provate pure FreeBSD o NetBSD... Esiste anche una scelta ancora piú esoterica: Hurd, il sistema operativo di GNU.
Del progetto GNU e della Free Software Foundation parleremo presto in un altro articolo; in questa sede basti dire che lo Hurd, pur avendo un'età paragonabile a quella di Linux, ha un grado di maturità molto inferiore. Tecnicamente, il progetto dello Hurd è molto piú moderno e sofisticato, prevedendo un'architettura a microkernel e la maggior parte dei servizi disponibili come moduli che girano in memoria non privilegiata; il suo stato di avanzamento è comunque ancora molto povero, presumibilmente a causa del differente modello di sviluppo adottato dalla FSF rispetto a Linus (v. articolo seguente). Lo Hurd è considerato ancora in beta, e non è quindi consigliabile ai deboli di cuore.

Il pinguino che creò il bazar

La storia di Linux nella prospettiva del modello "bazar", adottato dalla Netscape per lo sviluppo dei progetti nati dal rilascio pubblico dei sorgenti di Communicator

Nel 1997 Eric Raymond, un programmatore che aveva contribuito a vari programmi del progetto GNU, presentò al Linux Kongress di Würzburg un articolo dal titolo "La cattedrale e il bazar", in cui analizzava il modello di sviluppo di Linux. Anche chi non conosce questo articolo –non molto diffuso al di fuori del mondo dell'Open Source– dovrebbe intuire che il bazar, ovviamente, è Linux: cosa c'è di piú "sovversivo", per usare lo stesso termine adottato da Raymond, di un sistema operativo sviluppato in meno di un anno praticamente da una sola persona, e che raggiunge la maturità, in meno di due anni, per mano di una quantità impensabile di programmatori e utenti sparsi in tutto il mondo? Secondo il modo di pensare corrente, sostenuto da tutti i libri di ingegneria del software, i progetti di una certa complessità vanno progettati da una piccola comunità di programmatori che lavorino a stretto contatto, e accuratamente verificati anche prima del rilascio in beta come una cattedrale, appunto. Com'è possibile che un progetto della complessità di un kernel Unix possa venir sviluppato in un tempo così breve e con una metodologia così sovversiva?

L'inizio di un mito

Linus Torvalds acquistò un 386 nel gennaio 1991, periodo in cui queste macchine erano divenute sufficientemente economiche per essere acquistate da uno studente. Immagino che, come molte altre macchine analoghe, questo PC montasse MS-DOS, ma solo un piccolo numero di utenti utilizzava anche Minix, il sistema operativo didattico Unix-oide sviluppato da Andrew Tanenbaum e collaboratori. Poco impressionato dalle caratteristiche di Minix, Linus iniziò a sviluppare programmi che utilizzassero direttamente l'hardware della macchina, senza l'intermediazione di alcun sistema operativo, caricati direttamente dal processore al momento del reboot: un altro utilizzo abbastanza atipico di un computer, ma evidentemente il giovane programmatore aveva già abbastanza chiaro il divario tra le effettive possibilità dell'hardware che possedeva e l'utilizzo che ne facevano i sistemi operativi disponibili. All'inizio del mese di luglio, in un messaggio sul gruppo *comp.os.minix*, Linus nominava con aria indifferente un "progettino" cui stava lavorando: qualcuno si sarà forse chiesto a cosa potesse servirgli lo standard POSIX (che specifica le funzioni di interfaccia

Link

http://www.mozilla.org/
Sito del progetto Mozilla.

http://www.kernel.org/
Archivio di nuove e vecchie versioni di Linux.

http://www.tuxedo.org/~esr/writings/cathedral-bazaar/
L'articolo originale di Eric S. Raymond, "The Cathedral and the Bazaar".

che un sistema Unix deve offrire alle applicazioni), e la risposta arrivò dopo soli due mesi: Sto lavorando adun sistema operativo libero (solo un hobby, non sarà grande e professionale come gnu) per 386/486... vorrei sapere quali caratteristiche la maggior parte delle persone desidera. L'entusiasmo tra i lettori del newsgroup, tutti hacker appassionati, fu immediato: domande tecniche e richieste di effettuare il test beta del nuovo sistema abbondavano, e a metà settembre la prima versione del nuovo sistema operativo era disponibile su un server ftp, per coloro che ne conoscevano l'esistenza. Linus non era ancora soddisfatto a sufficienza per potere annunciare il rilascio al newsgroup, cosí, si limitò ad annunciare la disponibilità via mail a poche persone selezionate. Con una certa arbitrarietà, Linux Today (http://www.linuxtoday.com) ha "stabilito" che la data di nascita di Linux corrisponde a quella della creazione dell'archivio della prima versione: il 17 settembre 1991. Il 5 ottobre, il sistema era sufficientemente maturo per essere annunciato pubblicamente: I sorgenti di questo mio progettino si trovano su *nic.funet.fi* (128.214.6. 100) nella directory */pub/OS/Linux* (il primo annuncio pubblico del nome del sistema operativo, sebbene sottinteso nel nome della directory!). Un altro brano di quel primo annuncio è estremamente interessante: Posso quasi sentirvi chiedere, "Perché? Hurd sarà disponibile tra un anno (o tra due, o il mese prossimo, chissà), e già utilizzo Minix". Questo è un programma scritto da un hacker per gli altri hacker. Innanzitutto, va notata la poca fiducia nella prossima disponibilità dello Hurd: una previsione azzeccata, dal momento che a tutt'oggi sono disponibili solo versioni beta, e io personalmente sospetto che la Free Software Foundation non abbandoni il progetto solo per motivi di immagine. In secondo luogo è interessante il modo in cui Linus evitava il confronto del suo sistema operativo con Minix: solo pochi mesi dopo, tra gennaio e febbraio '92, sullo stesso newsgroup avrebbe avuto luogo un'accesa discussione sui meriti relativi dei due sistemi (ne parleremo in un prossimo articolo), e in quell'occasione il tono sarebbe stato ben diverso. Infine, va notato il risalto dato alla cultura hacker: lo scopo principale di quei mesi di studio era, nelle intenzioni dell'autore, sviscerare le potenzialità del 386: chi avesse studiato il suo codice avrebbe imparato l'architettura del processore meglio e piú rapidamente che in qualunque altro modo. Anche le release notes della versione 0.0.1 sono estremamente interessanti. L'autore indica chiaramente la logica del progetto: un efficiente utilizzo delle caratteristiche hardware del 386, un'architettura estremamente tradizionale ma la più efficiente possibile, e allo stesso tempo, sufficientemente semplice da essere implementata il più rapidamente possibile. Otto mesi sono un tempo talmente breve da lasciare di stucco. C'è una sezione che merita di essere citata per intero:

7. Scuse :–)

Questo non è ancora "la madre di tutti i sistemi operativi", e chiunque spera in questo dovrà aspettare la prima vera release (1.0), e anche allora potrebbe continuare a preferire minix. Questa è una release per le persone interessate a vedere che aspetto abbia Linux, e non c'è ancora un vero supporto. Chiunque abbia domande o suggerimenti (anche bug-report se decidete di farlo funzionare sul vostro sistema) è incoraggiato a contattarmi via mail. La versione 1.0 sarebbe venuta dopo piú di due anni, ma Linus già sapeva che ci sarebbe stata. Modestia e speranza di successo sono, come in tutti i suoi scritti, perfettamente bilanciate. Ancora un punto degno di nota è il copyright: data la piccola dimensione del progetto, veniva utilizzata

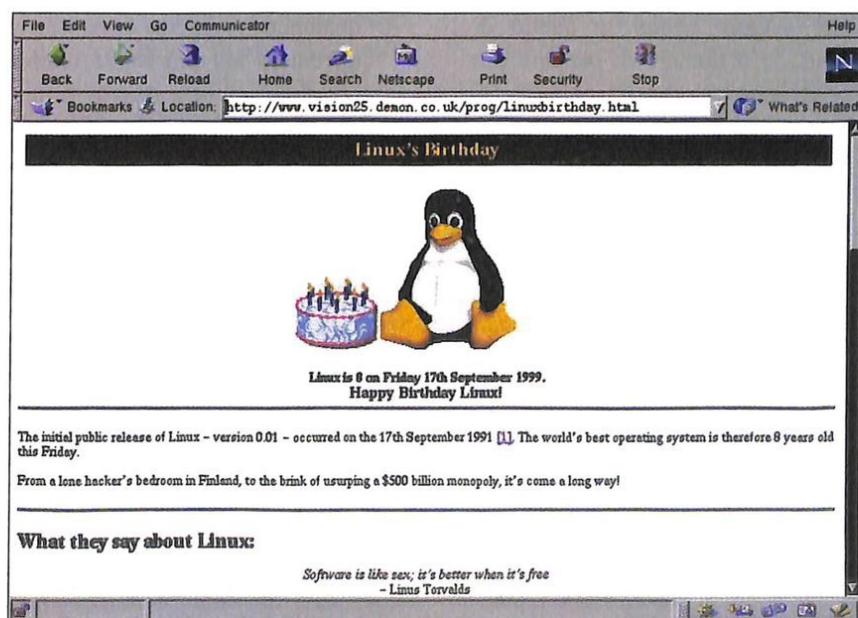



Figura 1 – L'articolo sull'ottavo compleanno di Linux, proposto dalla rivista on-line "Linux Today" (http://www.linuxtoday.com/). Simpatica, come al solito, la citazione di Linus.

una versione estremamente limitata della Licenza Pubblica GNU, con l'aggiunta del divieto di chiedere soldi per la distribuzione, anche solo per i costi dei dischetti o di spedizione. A partire dalla versione 0.12 (che in effetti era la sesta: i cambi dopo la 0.0.3 erano stati tali da giustificare il "salto" alla 0.10) sarebbe stata introdotta la GPL, con effetto a partire dal primo febbraio 1992. E questo avrebbe più avanti dato il via alle distribuzioni, prima fra tutte la storica Slackware, ancora in voga tra i "devoti".

Inizia il bazar

L'applicazione della GPL significava non solo l'apertura verso il "mondo esterno" all'ambiente degli hacker, ma anche la semplificazione della gestione del codice: le modifiche venivano semplicemente inserite conservando il Copyright di Linus Torvalds sotto la GPL, indipendentemente da chi ne fosse l'autore, anche perché si trattava ancora di piccoli contributi. Altrettanto importanti, e molto più numerosi, erano i bug report; in quelle prime versioni, la configurazione del sistema era un vero lavoro da hacker, e chi era in grado di effettuarla era in genere anche in grado di investigare, almeno fino a un certo punto, la ricerca del baco: a quel punto, spesso era piú proficuo che la correzione effettiva fosse effettuata da Linus, che conosceva il sistema meglio di chiunque altro ed era un vero maestro del debugging. Il numero di persone che lavorava al kernel, per perfezionarne gli aspetti o per scrivere dei driver per il proprio hardware, cresceva rapidamente; ma ancora nella versione 0.95 (un nuovo salto di qualità!) i nomi indicati esplicitamente nelle release notes erano pochissimi. Il vero bazar ebbe inizio con la versione 0.99, la quale aggiungeva un terzo livello di numerazione, ebbe un numero spropositato di release, dalla 0.99.1 fino alla 0.99.15c, e fu sviluppata per quasi due anni. Nella versione 1.0, rilasciata nel marzo del 1994, compariva per la prima volta un file CREDITS, contenente 80 nomi di sviluppatori (compreso quello di Linus, indicato come General kernel hacker) e il numero di beta-tester era dell'ordine delle centinaia. Con la versione 1.0 veniva introdotto anche il sistema di numerazione delle versioni attualmente in vigore; la 1.0 era considerata la prima versione "stabile", ma contemporaneamente ad essa venne rilasciata anche la 1.1, sulla quale lavoravano gli sviluppatori. Sulla 1.0, invece, veniva effettuato solo un lavoro di correzione di bachi. Quando la 1.1 fu considerata sufficientemente stabile, venne rilasciata la versione 1.2 e la versione di sviluppo divenne la 1.3; analogamente, al momento attuale la versione "stabile" è la 2.2 e il lavoro di sviluppo avviene sulla 2.3. I nomi riportati nei ringraziamenti sono divenuti 273 (sempre compreso quello di Linus), e il numero di persone che ha lavorato al progetto come beta-tester è dell'ordine delle migliaia. Il successo di Linux è stato determinato da un certo numero di fattori: la disponibilità di una piattaforma economica, popolare e potente, progettata tenendo conto dell'esperienza derivante da computer di fascia piú alta; l'ideale del software libero, trapiantato nella cultura hacker degli anni '90, che permetteva un progetto di sviluppo collettivo animato solo dal desiderio di portare avanti il sistema nella maniera migliore; un coordinatore di geniali capacità tecniche, in grado di sviluppare rapidamente il progetto fino allo stadio in cui fosse utilizzabile e quindi ci fosse un numero sufficiente di persone interessate a lavorarci. L'articolo di Eric Raymond ha avuto il merito di mettere in luce un altro importantissimo fattore di novità introdotto da Linus: un modello di sviluppo potente e originale, effettuare nuove release subito e spesso, delegare quanto piú possibile, mantenersi aperto fino alla promiscuità, per dirla con le sue stesse parole.

Questo modello di sviluppo costituisce un inserimento delle novità introdotte da Linus nel modello di sviluppo dell'Open Source; il processo funziona piú o meno come segue:

1. Il programmatore trova un problema che lo interessa,
2. poi cerca un programma "open source" esistente che si avvicini alla soluzione del suo problema,
3. quindi lo modifica fino a raggiungere uno stadio in cui altre persone sono interessate allo stesso programma, e fa il possibile per permettere loro di essere di aiuto per lo meno nel debugging;
4. qualunque problema deve essere affrontato nel piú breve tempo possibile, anche a scapito di una verifica accurata della soluzione trovata;
5. infatti, se si hanno abbastanza utenti, qualunque baco sarà ovvio per qualcuno!

Quest'ultimo punto sembra essere di particolare interesse: dati abbastanza occhi, qualunque baco è poco profondo. Raymond ha battezzato questa frase "la legge di Linus", in quanto sembra essere critica nel modello di sviluppo del bazar. Anche la caratterizzazione di questa idea data da Linus è interessante: Qualcuno trova un baco, e qualcun altro lo capisce. E in generale la

cosa piú difficile è trovarlo. Non ho riportato tutte le idee di Raymond, ma solo il nocciolo che costituisce la teoria del modello di sviluppo di Linux; è interessante osservare che l'idea di una larga base di utenti che partecipi quanto piú possibile allo sviluppo è nuova non solo rispetto alla tradizionale ingegneria del software ma anche rispetto al mondo dell'Open Source. In questo ambito, un hacker poteva riutilizzare i programmi di altri, esplorare nuove linee di sviluppo, contattare l'autore per problemi o suggerimenti, ma sempre in fasi molto avanzate del progetto, dopo un rilascio di una versione stabile, e certamente non nel corso dello sviluppo iniziale. L'innovazione fondamentale di Linux è stata quella di avere quanti piú utenti possibili non appena il programma fosse in uno stadio accettabile. È pure interessante notare che Raymond sembra quasi voler minimizzare le capacità tecniche di Linus (pur non potendole ignorare del tutto...), dando ad intendere che il modello di sviluppo è anche piú importante delle capacità tecniche; a riprova di questo egli descrive l'applicazione di questa teoria utilizzata da lui stesso nello sviluppo di fetchmail, un programma di recupero della posta elettronica presente in tutte le distribuzioni di Linux. Il suo scopo è quello di dimostrare che il modello di sviluppo non dipende da qualche geniale e inimitabile capacità di Linus Torvalds, ma è utilizzabile da tutti in qualunque contesto; e nel tentativo di dimostrare questa tesi, arriva anche a falsare gli eventi, sostenendo che Linus non abbia scritto il suo sistema operativo da zero, ma abbia iniziato riciclando il codice di Minix. Questo sarebbe stato tra l'altro contrario al Copyright di Minix, che vieta la distribuzione di versioni personalizzate; e in ogni caso la strada seguita nello sviluppo, raccontata personalmente da Linus in una news del '92, riporta le prime fasi secondo le linee che ho riportato in precedenza. Le prime versioni di Linux utilizzavano lo stesso filesystem di Minix per semplicità, ma il codice sorgente era completamente nuovo, tanto che fin dall'inizio vi era il supporto per thread multipli. Questo non toglie molto alle tesi di Raymond, che restano sostanzialmente esatte; il ruolo del coordinatore, che è fondamentale per un progetto in stile bazar ben piú che per uno tradizionale, è stato leggermente tralasciato, fors'anche per modestia, dal momento che buona parte dell'articolo si incentra non su Linux ma sul tentativo dell'autore di imitarne consapevolmente lo stile. Il coordinatore deve essere in grado di offrire rapidamente un prototipo funzionante e deve rispondere rapidamente alle aspettative degli utenti, se vuole preservarne l'entusiasmo; e deve avere un buon talento di progettazione e buone capacità di debugging. Piú complesso è il progetto, ovviamente, piú importanti diventano queste capacità; ben poche persone sarebbero state in grado di programmare il kernel di un sistema operativo in meno di un anno, e di continuare a seguirne lo sviluppo anche durante i successivi enormi ampliamenti.

Il dinosauro e il bazar

Credo che Raymond abbia trascurato anche un secondo aspetto, per illustrare il quale avrò bisogno di una piccola digressione. La maggior parte dei lettori ricorderà "la guerra dei browser", il periodo in cui Microsoft e la Netscape Corporation si contendevano il mercato dei browser giocando al ribasso; ma Netscape Navigator era già completamente gratuito, cosí la Microsoft non trovò di meglio per vincere che inserire il proprio browser direttamente nel sistema operativo, risparmiando cosí all'utente (o al rivenditore) anche la fatica di un'installazione in piú. A quel punto la Netscape, il cui guadagno proveniente dal browser era per ovvi motivi estremamente limitato, decise di rendere pubblico il codice di Netscape Communicator (l'estensione di Navigator). Stiamo parlando del mese di gennaio 1998, quindi meno di due anni fa; il codice di Navigator fu reso pubblico su Internet il 31 marzo. Nel frattempo Raymond, come egli stesso racconta in un epilogo aggiunto all'articolo che abbiamo discusso, era stato contattato dalla Netscape come consulente per l'avvio del loro progetto. Ma quale progetto? È stata la mia reazione quando ho letto l'articolo, se stavano dando via il codice? Il progetto era naturalmente quello di continuare lo sviluppo; e la posta in gioco era quella di introdurre il modello di sviluppo del bazar nel mondo commerciale. Per dirla con le parole di Raymond: Netscape sta per fornirci un realistico test su larga scala del modello del bazar nel mondo commerciale. La cultura dell'Open Source è di fronte a un pericolo; se il lavoro della Netscape non funziona, il concetto dell'Open Source sarà discreditato al punto che il mondo commerciale non vi si riavvicinerà per un'altra decade. Nel leggere queste parole, ho sempre avuto una sensazione sgradevole; quale può mai essere il pericolo? Quello che non sia possibile fare soldi con l'open source? Ma il suo scopo principale non è mai stato commerciale ma, piuttosto idealistico: sviluppare software di qualità disponibile per

tutti. Che poi sia possibile commerciare in software libero, soprattutto a livello di supporto, è un altro discorso; ma lo sviluppo di software open source da parte di produttori di software commerciale è di per sé fonte di contraddizioni. Un esempio chiarificatore è narrato da Richard Stallman in uno dei suoi "saggi filosofici"; egli racconta di un rappresentante di una casa di software che, a un convegno, disse qualcosa del tipo: "Non abbiamo la minima intenzione di rendere disponibili i nostri prodotti come open source, ma forse potremmo considerare una specie di open source "interno"; se permettiamo alle aziende che commerciano il nostro software di guardare il codice, potrebbero correggere i bug per i clienti, e saremmo in condizioni di fornire un servizio migliore". L'argomento è corretto e porta sicuramente acqua al mulino dell'open source, ma non si basa certo sui principi idealistici che hanno ispirato Stallman e i suoi collaboratori. Anche la Netscape, com'era lecito aspettarsi, ha seguito una strategia analoga: forse il mondo dell'open source può aiutarci ad ampliare la nostra gamma di prodotti e a renderla piú efficiente, si sono detti; non potremo venderli, certo, ma potremo vendere il supporto, che rende anche di piú. A distanza di diciotto mesi dalla liberalizzazione del codice di Communicator, solo una organizzazione ne sta utilizzando il codice: Mozilla, sponsorizzata dalla stessa Netscape, e i risultati sembrano essere non esaltanti. I prodotti sono ancora in fase di test beta, e non sembra esserci grande interesse nel loro utilizzo. Diciotto mesi non sembrano molti, ma lo diventano se vengono paragonati con i tempi di sviluppo di Linux; cos'è successo, dunque? Il pericolo indicato da Raymond è diventato realtà?

Ma qual è dunque questo pericolo? Il modello del bazar sembra funzionare, e tra l'altro sta ottenendo grandi successi con lo sviluppo dell'ambiente di desktop Gnome, che ha un numero di sviluppatori impressionante; perché non ha funzionato con Mozilla? Non conoscendo i dettagli del progetto, posso immaginare che il problema sia stato un altro dei punti critici del modello del bazar: un insufficiente numero di utenti. I progetti realizzabili con Navigator sono molto legati agli aspetti piú commerciali di Internet e poco interessanti per un hacker; allo stesso tempo sono anche molto ampi e ambiziosi, di grande complessità. La versione attuale di Linux è indubbiamente di una complessità estrema: però ha 237 sviluppatori attivi, oltre ad un numero di beta tester proporzionato. I progetti di Mozilla, nel loro complesso, richiederebbero numeri analoghi; numeri che è difficile trovare per quel tipo di progetto tra gli hacker.

Conclusioni

Il modello del bazar spiega bene lo sviluppo di Linux, ma non è di universale applicabilità, in quanto richiede un adeguato numero di utenti molto interessati al progetto e con un buon bagaglio tecnico. Come tale, esso funziona molto bene per il software di supporto alla programmazione sistemi operativi, compilatori, librerie, gestione della rete, ... ma diventa difficilmente utilizzabile per sviluppare programmi applicativi. Questo spiega anche perché i pacchetti di produttività d'ufficio esistenti per Linux sono commerciali: nessun programmatore sarà sufficientemente interessato a un programma di videoscrittura da perdere tempo e fatica a capirlo ed effettuarne il debugging, a meno che, naturalmente, non sia pagato per farlo. Il software commerciale è quindi, a mio avviso, ancora una componente fondamentale non solo dell'ambiente Windows ma anche di Linux, almeno fino a che non venga intrapresa la strada dei programmi applicativi che vadano incontro, non solo alle esigenze dell'utente inesperto ma, anche del programmatore appassionato.

Francesco Marchetti Stasi

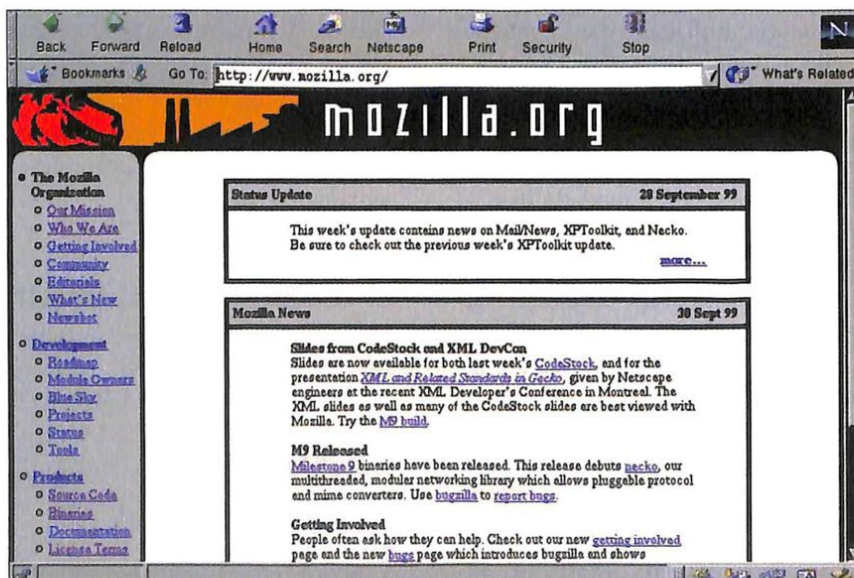



Figura 2 – La "home page" del progetto Mozilla. http://www.mozilla.org/.

La controversia Corel

Corel e GPL, una bolla di sapone o una violazione dei diritti d'autore?

Tutto è cominciato a San Jose in California nell'estate di quest'anno, esattamente il 10 agosto 1999, quando la Corel Corporation ha rivelato al mondo informatico che era pronta a rilasciare una nuova distribuzione del sistema operativo Linux. COREL Linux, questo il nome della nuova distribuzione, si baserà sulla conosciutissima ed amatissima, almeno dal mondo accademico nostrano, Debian/ GNU; l'interfaccia grafica adottata sarà la KDE (K Desktop Environment).
Sempre nell'annuncio di agosto, la Corel ha notificato la disponibilità, per i primi di settembre '99, di una versione di prova per tutti i beta tester del mondo Linux, e un rilascio definitivo entro la fine dell'anno.
Alla distribuzione, però, ha anche allegato un'intesa che nega la possibilità di distribuire liberamente il software in beta testing, scatenando una tremenda tempesta di proteste.
Non sembrerebbe neanche una cosa tanto blasfema, abituati come siamo a certe grandi società di software che fanno pagare anche la licenza per accedere ad un server già regolarmente registrato e pagato; ma in questo caso c'è di mezzo una licenza un po' particolare, la cosiddetta GPL (General Public License) del progetto GNU.

GPL e la Free Software Foundation

Per capire cosa è la GPL, bisogna risalire ad un altro concetto: il Free Software.
Spiegare il FreeSoftware vuol dire parlare di "copyleft". Tutti questi meccanismi sono stati pensati dalla Free Software Foundation, un'associazione fondata da Richard Stallman del MIT, che si occupa dell'abolizione dei diritti d'esclusi-

Riferimenti Bibliografici

- **HACKERS**
  Leavy, Steven
  (Shake Edizioni Underground)
  La vera storia degli hacker, dai primi anni '60 ad oggi.

- **No Copyright**
  Nuovi diritti nel 2000
  Valvola Scelsi, Raf
  (Shake Edizioni Underground)
  10/94
  Traduzione e presentazione di una gran quantità di testi "sacri" sui nuovi diritti digitali.

va sul software, e della possibilità di rendere tutti gli utenti dei prodotti inerenti il progetto GNU (acronimo di 'GNU is Not Unix') liberi di ridistribuire e modificare il software.

Il copyleft è proprio questa libertà. Stallman ha voluto usare questo nuovo concetto perché ha capito che per combattere le leggi sul copyright bisogna ricorrere alle stesse armi legali con cui le grandi corporazioni di software proteggono i loro codici e le loro applicazioni.

Lo strano nome, copyleft, deriva proprio dal fatto che gli sviluppatori di software utilizzano il copyright per togliere agli utenti la libertà di usare, modificare e ridistribuire i loro programmi; Stallman, invece, usa il copyright per garantire queste libertà. Da ciò deriva la complementarietà della seconda parte del vocabolo in cui right diventa left.

Per capire lo stratagemma si può pensare al famoso proverbio: 'chi di spada ferisce, di spada perisce'. Applicare il copyleft ad un programma vuol dire: prima proteggerlo con un copyright e poi concedere una licenza che dia a chiunque il diritto di disporre del codice del programma, o di ogni programma derivato, ma solo se i termini della licenza non sono cambiati. In questo modo il codice e le libertà diventano legalmente inseparabili.

I termini della licenza di distribuzione sono contenuti nella General Public License (GPL). Ma di essa esiste anche una versione modificata che si utilizza in quasi tutte le librerie GNU: la Lesser General Public License (LGPL), già denominata Library GPL.

La vecchia volpe di Stallman, per non incorrere in problemi legali, che evidentemente conosce bene, ha tradotto la GPL in varie lingue, facendovi inserire un paragrafo iniziale sulle licenze tradotte in cui si specifica a chiare parole che la traduzione non ha valore ai fini legali. Viene il dubbio che il mondo forense americano sia anche più cavilloso dell'avvocato Azzeccagarbugli dei "Promessi sposi".

Il software che viene così rilasciato da chiunque lo desideri, sempre sotto licenza, può essere classificato come Free Software. Anche in questo caso Stallman è molto analitico sull'accezione delle parole, e, non a caso, usa il termine 'Free' con il significato di libertà e non di gratuità, e per evitare confusione con software di tipo freeware, shareware o anche solo gratis. In effetti, il concetto di Free Software va molto al di là di queste tipologie.

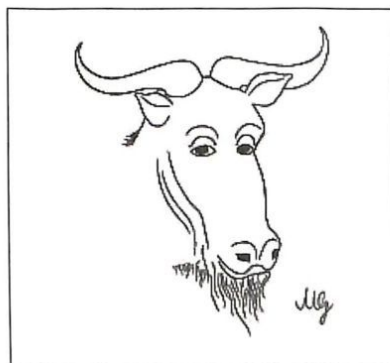



Fig. 1: Il logo di GNU

Il Free Software si riferisce alle libertà dell'utente di far girare, copiare, distribuire, studiare, modificare e migliorare il codice. Per la precisione, i livelli di libertà dell'utente sul software sono quattro:

- La libertà di utilizzare il programma per qualsiasi scopo (libertà 0)
- La libertà di studiare come lavora il programma e di adattarlo alle proprie esigenze (libertà 1)
- La libertà di ridistribuire delle copie per aiutare gli altri (libertà 2)
- La libertà di migliorare il programma e rilasciare pubblicamente le migliorie così che la comunità ne possa beneficiare (libertà 3)

Un programma è Free Software se gli utenti hanno tutte queste libertà.

Free Software vs Open Source

Non si può ignorare che il mondo dell'IT (Information Technology) sta subendo, in quest'ultimo periodo, il fascino di un nuovo tipo di filosofia di software, che è comunque presente già da qualche anno sulle scene informatiche, e che ha rilanciato anche Linux: il software Open Source.

L'Open Source è arrivato perfino nelle grandi aziende e sui tavoli della classe manageriale, magari anche solo attraverso le riviste specializzate; comunque, è entrato in santuari che, fino a poco tempo fa, non prestavano la minima attenzione a software che non provenisse da marchi blasonati o da multinazionali con strutture interne degne di un faraone.

A causa di tale diffusione si è cominciato, come per tutte le cose che non si conoscono, a parlare di tutti gli aspetti legati a questo argomento in modo confuso, usando indistintamente sia la parola Open Source che la parola Free Software.

Il rigido, almeno sulla semantica, Stallman ha quindi prontamente divulgato un documento sulla

distinzione dei due concetti che apparentemente possono sembrare uguali, ma che nascondono delle differenze quasi sostanziali a livello concettuale.
Open Software è un concetto ampio che dichiara solo che può essere visto il codice del programma, e quindi comprende, oltre a tutta la classe Free Software, anche altri software semi-liberi e alcuni addirittura proprietari con determinate forme di licenza.
Il Free Software quindi è un sottodominio del dominio Open Source.
Tra di essi esiste una differenza sottile: per il Free Software la FSF deve insegnare agli utenti qual'è la parola giusta per un unico concetto ben definito, mentre nell'Open Source esiste un solo termine che, però, ha varie sfumature ed una sola è quella giusta.
In realtà l'Open Source è il Free Software rivestito di marketing, un modo per renderlo più accettabile al mondo del business. Non per niente la comunità Open Source ha creato anche un marchio (OSI acronimo di Open Source Initiative) con cui si devono contrassegnare tutti i prodotti software che soddisfano alla licenza OS.
Stallman paragona i due schieramenti a due partiti politici della stessa comunità che differiscono sulle basi ideologiche, ma che si incontrano sulle raccomandazioni pratiche. Sul sito della comunità Open Source (www.opensource.org) si può leggere una definizione sintetica del software di questo tipo: "Open Source promuove un'analisi indipendente e una rapida evoluzione del codice sorgente. Per essere certificato OSI il software deve essere distribuito seguendo i termini di una licenza che garantisce liberamente i diritti di lettura, modifica, ridistribuzione ed uso del software".
Qualcuno attenta al diritto inviolabile della libertà?
La famosa società canadese di software Corel, fino ad ora, ha lavorato esclusivamente su prodotti commerciali di grande successo, tipo Corel Draw e WordPerfect. Per realizzare una distribuzione di Linux all'altezza dei software precedenti, ha dovuto quindi adottare un diverso tipo di approccio nella fase di test.
Precedentemente creava una piccola cerchia ristretta di programmatori che si occupavano di tutta la fase di test, ma, nel caso di Linux, il management della Corel si sarà chiesto chi avrebbe potuto testare la nuova distribuzione trovandone tutti i possibili bug e magari anche risolvendoli ad una velocità quasi prodigiosa.
La rapidità era un fattore critico anche perché l'annuncio e la fase di rilascio erano date molto vicine tra di loro. L'unica risposta logica a tale requisito è stata: la comunità Linux.
Purtroppo la Corel si è trovata in una situazione molto imbarazzante. Ha invitato i beta tester a lavorare sulla distribuzione per provarla con tutte le piattaforme e a migliorarne tutti gli eventuali problemi e punti deboli, ma, allo stesso tempo, ha espressamente detto che il software non doveva essere distribuito a nessun altro e che, anzi, doveva essere distrutto dopo 45 giorni dalla ricezione.
Queste postille sono state subito considerate come una violazione di parecchi articoli della GPL e la comunità è insorta in massa, chiaramente sempre in maniera virtuale. Ma questa virtualità non sempre è rimasta tale e talvolta è diventata anche reale, come quando è entrato in gioco Slashdot, un sito Web fonte inesauribile di notizie preziose per tutti gli amanti dell'informatica e seguito da migliaia di tecnici che desiderano scambiarsi informazioni.
Slashdot è un seguace della fede Free Software e quando si verificano delle violazioni alla licenza si scaglia contro "l'infedele" e porta con sé tutta la schiera di sostenitori della comunità.
Come in precedenti situazioni, anche in questo caso Slashdot ha funzionato da amplificatore per le proteste contro la Corel, costringendo la società a rilasciare comunicati, lettere e fax di giustificazione per i termini di testing incriminati. La Corel è dovuta ricorrere anche ad un'intervista su Linux World, altro sito on-line di grande interesse per il mondo Linux e Free Software, in cui spiega le sue ragioni e precisa che non ci sono state, e non ci saranno, violazioni alla libera distribuzione di software. La richiesta incriminata derivava dal solo fatto di non voler rilasciare la versione beta finché non fosse stata provata e consolidata. Il rilascio di software instabile e poco robusto avrebbe creato un calo di immagine disastroso della società canadese e la reputazione, per determinati ambienti commerciali, vale molto.

Conclusioni

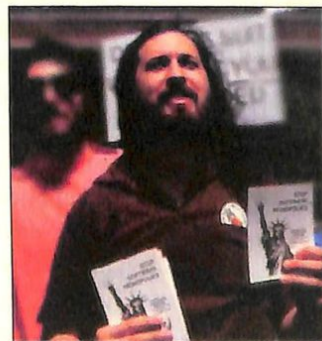
In questo caso si può ragionevolmente credere che la Corel Corporation sia in buona fede ma, non è detto che altre società non prendano iniziative lucrose sulla base di software libero. GPL e OSI sono due realtà diverse di una stessa visione: il software deve

essere liberamente utilizzato, distribuito e modificato; i miglioramenti del codice ridistribuito andranno a beneficio di tutta la comunità informatica. Non è altro che la vecchia teoria dell'evoluzione naturale: il più adatto alla fine vince e si moltiplica. La specie si evolve in maniera sempre più perfetta. Il commercio e il profitto non devono assolutamente entrare in questo contesto e, applicare le leggi del mercato alle filosofie hacker della FSF significa stravolgerne completamente il significato, ed andare contro tutti i principi che hanno origine dal MIT degli anni '60, e dai geni del computer, che hanno fatto la storia dell'informatica.

Marco Gastreghini

L'ultimo degli hacker

Richard Stallman entrò nel MIT (Massachusetts Institute of Technology) nel lontano (o forse non così lontano) 1971. La passione dell'hackeraggio nel tempio degli hacker americani lo invase da subito e non lo lasciò più. Appassionato di computer fin da ragazzo, entrò subito in un centro di calcolo a Manhattan, città dove era nato. Quando arrivò ad Harvard era già uno "smanettone" di alto livello e, forse proprio per questo, non resistette a lungo in quell'Università e finì quasi inevitabilmente al MIT. Al Tech Square, nel laboratorio di Intelligenza Artificiale, non c'erano ostacoli alla libera ricerca e all'hackeraggio puro, visto come essenza della vita, e Stallman non cercava altro. L'amministratore dell'Ia Lab lo assunse come programmatore sistemista e, nel frattempo, Stallman prese una laurea magna cum laude in fisica ad Harvard. Il laboratorio veniva visto da Richard come l'incarnazione dell'etica hacker, una sorta di anarchismo costruttivo.

Stallman, il cui nickname preferito era Rms, utilizzò la filosofia hacker per produrre la sua opera più conosciuta: Emacs, un editor di testo che poteva essere personalizzato in maniera illimitata. L'architettura aperta incoraggiava le persone a modificarlo aggiungendovi parti e migliorando quelle già presenti. L'autore distribuiva gratuitamente il software alla condizione di rendere disponibili tutte le estensioni apportate (un embrione di Free Software) e Emacs divenne in breve tempo l'editor di testi standard nei dipartimenti universitari di informatica. Rms odiava le password e faceva in modo che sui computer da lui gestiti non ve ne fossero. Quando il MIT decise di assegnare delle password agli utenti registrati del laboratorio, tagliando quindi fuori tutti gli altri, Stallman chiedeva agli utenti registrati di inserire password nulle e dare la possibilità a tutti gli altri di accedere ai sistemi. Arrivò anche a decrittare il file delle password e ad inserire messaggi di login che riportavano il testo seguente: "Vedo che hai scelto la password <password>. Ti consiglio di cambiarla con <return>. E' più semplice da usare e sostiene il principio che non ci dovrebbero essere password". Rms rimase all'Ia Lab fino a che non vide svanire tutta l'etica hacker che aveva fatto la storia dell'informatica così come ora la viviamo. Considerandosi l'ultimo vero hacker, ha sempre lottato per far tornare la purezza della magia del MIT contro lo sfruttamento delle società che, con il segreto industriale, avevano snaturato l'elemento chiave della sua etica: il libero flusso dell'informazione. Stallman lasciò il MIT con il progetto di scrivere una versione di Unix da distribuire liberamente e combatte ancora oggi: il progetto GNU ne è una prova più che tangibile. Tutto quello che fa lo riconduce ad un unico scopo: andare contro la proprietà del software che, come definisce lo stesso Stallman, è la sifilide del territorio digitale.

Una panoramica sul sistema Unix

Già prima della nascita di MS-DOS e Windows, Unix aveva una ampia base installata. Nei suoi primi 30 anni di vita si è notevolmente evoluto, pur rimanendo coerente al progetto originale dei suoi creatori. Una gran quantità di ottime distribuzioni commerciali, e un sempre più vasto parco di distribuzioni di pubblico dominio, una alternativa praticabile come sistema operativo per home o personal computer.

I moderni sistemi operativi sono costituiti da tre elementi di base:

- un nucleo (in inglese kernel) che ha la funzione di coordinare i processi applicativi, attraverso l'esecuzione alternata dei vari processi in attesa, assegnando in esclusiva a ciascuno l'uso della CPU, e di permettere l'efficiente ed ordinato accesso alle risorse condivise, come la CPU, la memoria, i dischi e i dispositivi esterni;
- un insieme di programmi di supporto e di applicativi di base. I programmi di supporto agevolano la soluzione dei problemi di amministrazione del sistema, o comunque svolgono un compito ausiliario e limitato rispetto ai programmi applicati, che forniscono in maniera diretta soluzioni ai problemi utente. Ovviamente, la suddivisione è in qualche modo sfumata ma, rimane comunque valida. Tra gli applicativi, alcuni sono compresi direttamente nel sistema operativo per le funzioni di grande utilità che svolgono, come l'editor visuale vi(1), mentre la maggior parte può essere installata in tempi successivi, in relazione alle specifiche esigenze dell'utente, come ad esempio il sistema di impaginazione di documenti TeX;
- infine una struttura efficiente di archiviazione e di gestione dell'accesso ai file (in inglese filesystem).

In Unix, il kernel è molto ridotto, esegue solo le funzioni essenziali di più basso livello per la temporizzazione dei processi, per la gestione dei file e il controllo dei dispositivi hardware.
Essendo il kernel così limitato, è necessario fare un grande uso dei piccoli programmi di supporto-programmi di supporto o di utilità, concepiti come strumenti software (comunemente detti tool).
L'uso di tanti piccoli tool, invece di un unico programma che esegue la totalità delle funzioni del sistema operativo, è una delle peculiarità specifiche di Unix. La connessione tra il kernel e i programmi di supporto è resa possibile dalla presenza di una libreria di funzioni di basso livello dette "chiamate di sistema", e di un'interfaccia per i dispositivi hardware che trova posto all'interno della normale gerarchia di file, attraverso l'uso di file speciali.

I Comandi

Come esempio, si riportano alcuni programmi di supporto che hanno un ruolo rilevante nelle operazioni del sistema Unix (avendo sempre come riferimento il Linux della distribuzione Red Hat 6.0):

Programmi di inizializzazione del sistema

• *init(8)* Il programma di attivazione dei processi.

• *getty(1)* Gestisce la fase di collegamento con un terminale, locale o remoto, per accettare la login utente.

• *login(1)* Permette il login utente per l'autenticazione con il sistema.

• *stty(1)* Codifica le caratteristiche del terminale.

Gestione del File-System

• *mkfs(8)* Costruisce le strutture base di un file-syste

• *mknod(1)* Crea un file speciale.

• *fsck(8)* Verifica lo stato di un file-systemfile-system.

• *mount(8)* Attiva o disattiva un file-systemfile-system • *sync(1)* Scrive i blocchi di dati presenti ancora nelle cache della memoria sul relativo file-system.

Gestione dei Processi

• *nice(1)* Modifica la priorità dei processi.

• *ps(1)* Visualizza lo stato dei processi.

Questi sono solo alcuni tra i tanti programmi di utilità che permettono, all'amministratore di un sistema Unix (chiamato solitamente 'utente rootroot'), di gestire correttamente tutte le operazioni necessarie a tenere in buon ordine il proprio sistema.

File-system

Il dischi di Linux, come quelli di MS-DOS, sono organizzati in una struttura gerarchica di file. Il sistema di archiviazione è basato su una struttura ad albero rovesciato (vedi Figura 1) la cui radice è la directory indicata dal carattere '/' e chiamata root, da non confondere però con l'utente amministratore del sistema che assume lo stesso nome, né con la sua directory di lavoro che si chiama proprio /root. Tutte le altre directory sono indicate a partire dalla root. Le uniche differenze con il DOS riguardano il carattere di separazione delle directory che è lo slash '/' invece del backslash '\', l'assenza del concetto di "lettera del drive" (ad es. C:), poiché i diversi dischi fisici o le partizioni sono inserite nella struttura normale delle directory e, come vedremo, si prendono carico di un intero ramo dell'albero sottostante. Nella struttura gerarchica trovano posto tutte le risorse accessibili dal sistema. Anche oggetti che non sono propriamente file possono comparire nel file-system. Ad esempio, i dispositivi di input/output, come le porte parallele o seriali, hanno un file speciale che le rappresenta all'interno della directory '/dev' (rispettivamente /dev/lp1 e /dev/ttyS0). Esistono quindi diversi tipi di file, tra cui:

File ordinari: programmi, dati, file di testo, script, etc.

Directory: raccolgono altri file creando una struttura gerarchica.

Collegamenti (Link): rappresentano lo stesso file con nomi diversi, eventualmente in directory diverse. Possono riferirsi a qualunque tipo di file, quindi anche a dispositivi e directory.

Devices: rappresentano i dispositivi hardware del sistema (schermo, floppy, HD, CD, etc.).

File FIFO: questi file permettono a processi indipendenti di comunicare tra loro.

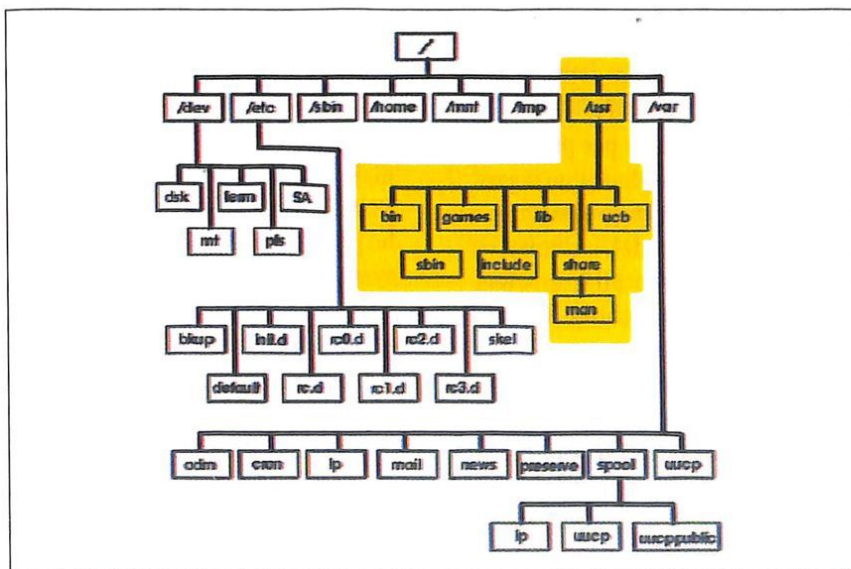



Figura 1 – La struttura gerarchica del file-system Unix. La zona gialla attorno alla directory /usr indica che l'intero albero, a partire da quel punto, è ospitato in una differente partizione del disco.

Esiste uno standard de-facto per l'organizzazione delle directory in un sistema Unix ma, non tutti i produttori vi si attengono scrupolosamente. Ogni dialetto Unix ha le sue particolarità, e districarsi correttamente tra tali differenze è il prezzo da pagare per il vantaggio di avere tanti produttori in concorrenza. Comunque, le distribuzioni Linux stanno convergendo verso una disposizione delle directory comune, che è anche quella adottata dalla RedHat 6.0 in cui:

/dev contiene tutti i file relativi ai devices.

/home é la directory base degli

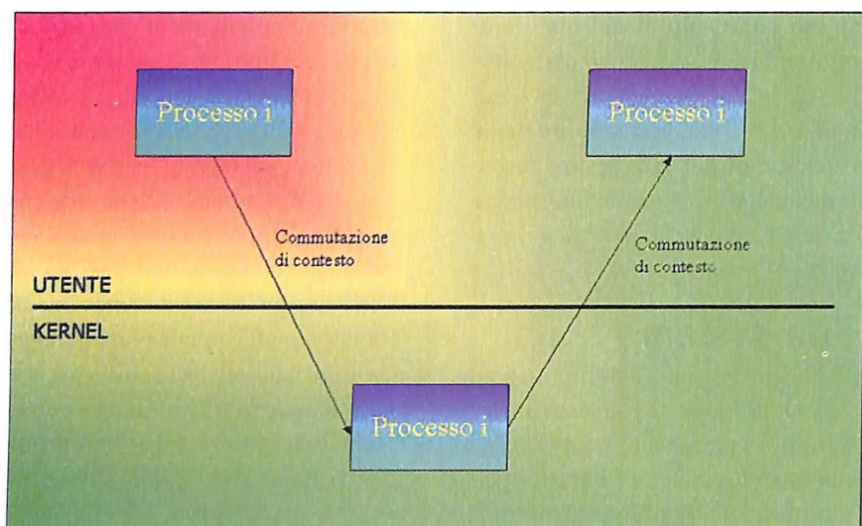



Figura 2 – Commutazione di contesto da modo utente a modo supervisore a causa dell'esecuzione di una chiamata di sistema. Un processo applicativo in esecuzione in modalità utente che richiede i servizi del kernel attraverso una chiamata di sistema. E' possibile vedere una commutazione di contesto e l'esecuzione delle istruzioni del kernel relative alla chiamata eseguite in modalità supervisore. Al termine della chiamata una nuova commutazione di contesto verso il modo utente permette al processo utente di proseguire dopo aver ricevuto i servizi della chiamata di sistema richiesta.

account degli utenti o delle applicazioni, ad esempio la directory di lavoro dell'utente exedre è /home/exedre.

/root é la directory di lavoro dell'amministratore di sistema, tenuta separata dagli altri utenti per ragioni di sicurezza, da non confondere con la directory root del file-system (ovvero '/').

/var é la directory dei file temporanei, dei messaggi di sistema, degli spool di stampa e dei file di log.

/bin contiene i principali comandi di sistema.

/sbin contiene i comandi di sistema assoggettati ad alcune speciali regole di sicurezza per evitare abusi.

/mnt contiene le directory utili per accedere ai dispositivi removibili (floppy o cd) e ai dischi non Linux.

/boot é presente in alcuni sistemi per contenere il kernel, altri sistemi usano direttamente la directory radice.

/etc é la directory contenente i file di configurazione delle applicazioni e del sistema operativo.

/tmp contiene file temporanei che vengono cancellati al boot del sistema.

/lib é la directory delle librerie usate dai comandi di base del sistema; contiene anche i moduli aggiuntivi non compilati nel kernel.

/proc é una directory virtuale (non esiste sul disco) che riporta il contenuto della memoria centrale del sistema in esecuzione, sia a livello globale, sia processo per processo.

/usr contiene le applicazioni, i programmi utente, la documentazione e tutte le librerie del sistema; ha una struttura simile alla directory radice, con delle sottodirectory bin, sbin, lib e temp.

La presenza di più dispositivi fisici di memorizzazione, o di più partizioni logiche all'interno di uno stesso dispositivo fisico, viene gestita attraverso la creazione di file-system indipendenti, eventualmente anche di diverso tipo, che sono "montati" all'interno della struttura gerarchica principale. Come mostra la Figura 1, per esempio, alla directory /usr è associata una partizione del disco diversa da quella della directory root. Tutte le operazioni che coinvolgono file presenti al di sotto di quella directory coinvolgeranno quella partizione del disco. Lo stesso meccanismo è valido per avere accesso a dispositivi rimovibili, come floppy disk o cd-rom, che vengono "montati" all'interno della normale gerarchia del sistema, nel caso sotto le speciali directory /mnt/floppy e /mnt/ cdrom rispettivamente. Ogni partizione del disco montata in una locazione diversa del file-system può anche avere una particolare directory denominata lost+found, che contiene il risultato delle operazioni automatiche di recupero di file eventualmente perduti da blocchi non preventivati del sistema. Dovendosi adattare a supporti fisici di memorizzazione, e a sistemi di registrazione eterogenei, il file-system di Unix si basa su un formato fisso, sempre costituito dalle seguenti quattro sezioni:

BOOT BLOCK
che contiene il codice macchina del "caricatore" del kernel per l'inizializzazione del sistema;
SUPER BLOCK che contiene tutte le informazioni peculiari del file-system, ad es. :

- dimensione del file-system;
- dimensione della lista che contiene le informazioni sui file presenti nel file-system (denominata i-list);
- numero di blocchi utilizzabili per aggiungere dati al file-system;
- data di modifica del SUPER

BLOCK;
- tipo di file-system;

I-LIST
che è la lista che contiene le strutture che definiscono le proprietà di ogni singolo file. Tale struttura è denominata i-node;
DATA BLOCK
contengono i dati reali del file-system, ovvero tutti i file, oltre che allo spazio vuoto;

Kernel

Il kernel controlla l'attività dei processi in esecuzione e risolve i conflitti di assegnazione delle risorse. Tra le risorse gestite dal kernel, certamente la più importante è la CPU, ovvero l'unità di esecuzione del flusso di istruzioni procedurali. Nei moderni personal computer, è presente solitamente una sola CPU, in grado di mandare in esecuzione una grande quantità di processi riservando contemporaneamente, a ciascuno, un breve intervallo di tempo d'esecuzione. Esistono versioni di Unix adatte a schedulare l'esecuzione, anche in sistemi contenenti più microprocessori, ed anche in Linux esiste una apposita estensione (denominata SMP). Il comportamento del kernel, rispetto alla gestione dei processi, determina le condizioni per una maggiore o minore tendenza al lavoro contemporaneo di più processi, e per la sicurezza del funzionamento dell'intero sistema. Senza scendere nei dettagli sulle differenze tra i vari tipi di gestione dei processi, va detto che Unix rientra tradizionalmente in quella classe di sistemi operativi denominati multitasking pre-empitive, ovvero sistemi che, con una estrema semplificazione, sottraggono forzatamente al processo in esecuzione il diritto di utilizzare la CPU dopo un periodo di tempo definito. In altri sistemi, al contrario, il processo deve essere per così dire "ben disposto" a cedere il controllo della CPU (in quel caso si parla multitasking cooperativo). Il grosso vantaggio dell'approccio pre-empitive è che risulta impossibile che l'intero sistema si blocchi per il malfunzionamento di un unico processo che, conservando il diritto di esecuzione, non libera mai la CPU per gli altri utenti. Il funzionamento dell'algoritmo di "schedulazione dei processi", ovvero il primo compito fondamentale del kernel, si basa su un valore di priorità di esecuzione, assegnato ad ogni processo, modificabile dinamicamente dal kernel a seconda dell'utilizzo di CPU. Una volta scelto un processo da mandare in esecuzione in base alla sua priorità, il kernel attende che si verifichi almeno una di queste condizioni:

- il processo esaurisce il proprio tempo di esecuzione;
- il processo entra in uno stato d'attesa;
- un processo a priorità più elevata irrompe sulla scena a causa dell'arrivo di un evento di cui era in attesa.

Il kernel interrompe quindi il processo in esecuzione e assegna la CPU ad un altro processo. Questo meccanismo di esecuzione ha anche il vantaggio che, fintanto che un processo non è in esecuzione, non è necessaria la presenza in memoria centrale del suo codice o delle sue strutture dati. È possibile quindi mettere in esecuzione molti più processi di quanti la dimensione della memoria centrale del sistema permetterebbe, avendo l'accortezza di spostare quelli non in esecuzione, secondo un insieme di regole definite, verso una zona specifica del disco denominata swap.
Il secondo compito fondamentale del kernel è la risposta alle chiamate di sistema inoltrate dai processi. Per comprendere le operazioni svolte dal kernel, in questa attività, è necessario sottolineare che un microprocessore compatibile con il sistema operativo Unix, ovvero tutti i moderni microprocessori, può lavorare almeno in due diverse modalità operative: un modo utente, adeguata a processi applicativi e un modo supervisore, riservato esclusivamente al kernel. In modalità supervisore, i meccanismi di protezione della memoria e dei processi, messi a disposizione dal microprocessore, risultano disabilitati. In Unix, un qualsiasi processo applicativo non potrà mai passare ad eseguire istruzioni di supervisione, se non in maniera controllata dal kernel attraverso le chiamate di sistema. Nel caso in cui un processo effettui una chiamata di sistema, dovrà eseguire una speciale istruzione macchina che forza la CPU a modificare il proprio contesto dal modo utente ad un modo supervisore. Questo cambiamento è denominato commutazione di contesto.
Nella Figura 2 è mostrato un processo applicativo, in esecuzione in modalità utente, che richiede i servizi del kernel attraverso una chiamata di sistema che provoca una commutazione di contesto, e l'esecuzione delle istruzioni del kernel relative alla chiamata eseguite in modalità supervisore.
Al termine della chiamata, una nuova commutazione di contesto verso il modo utente permette al processo utente di proseguire dopo aver ricevuto i servizi della chia-

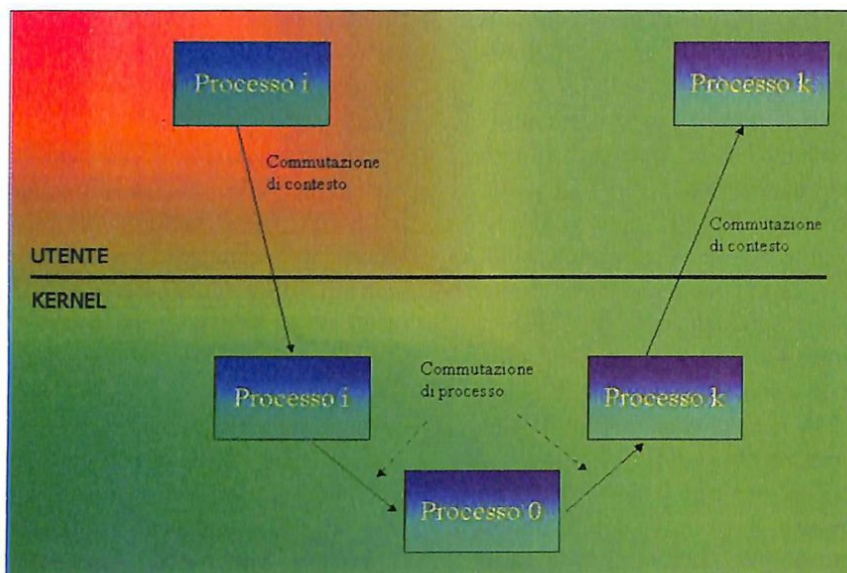



Figura 3 – Commutazione di contesto e di processo. Il processo in esecuzione (i) entra in wait. Avverrà, oltre alla commutazione di contesto relativa alla chiamata di sistema anche una commutazione di processo verso il kernel (processo con ID 0) che sceglierà di mettere in esecuzione un diverso processo (k).

mata di sistema richiesta. Come si può notare, una commutazione di contesto non provoca contemporaneamente una variazione del processo in esecuzione, ovvero l'ingresso in esecuzione di un processo diverso da quello chiamante. In alcuni casi, però, le chiamate di sistema richiedono che il processo chiamante sia posto in uno stato d'attesa di un evento esterno, ad es. la battitura di un tasto, o l'espirazione di un timer. In tal caso, si dice che il processo entra in wait. Avverrà, oltre alla commutazione di contesto, relativa alla chiamata di sistema, anche una commutazione di processo verso il kernel (processo con ID 0) che sceglierà di mettere in esecuzione un diverso processo (vedi Figura 3).

Il terzo ed ultimo compito caratteristico del kernel è la supervisione degli accessi ai file presenti nel file-system, per prevenire casi di creazione di incongruità tra la memoria centrale e quella secondaria, attraverso sovrascritture indesiderate degli stessi blocchi di dati, o eliminazioni di file che nel frattempo risultano usati dai processi attivi.

Inizializzazione del sistema

Durante la procedura di bootstrap, ovvero all'accensione o dopo un reset della macchina, viene ricopiato in RAM il kernel, che dalla procedura di installazione è di solito memorizzato in un file denominato vmlinux (o vmlinuz) presente nella directory principale o nella directory /boot. Il kernel è attivato automaticamente come primo processo del sistema. Un processo è un programma che, presente in memoria principale, può essere messo direttamente in esecuzione dal sistema.

Quando il kernel è appena attivato, ed è cioé il primo ed unico processo attivo, inizializza le strutture dati per la gestione del sistema e il collegamento con i dispositivi di I/O. Una delle strutture dati più importanti create durante il bootstrap ed aggiornata dal kernel è la tabella dei processi, che contiene tutte le informazioni su ogni processo finché è attivo nel sistema. Ovviamente, al kernel tocca la prima locazione nella tabella dei processi (ID 0). Alla creazione di tutti gli altri processi non provvede direttamente il kernel ma il programma d'utilità init(8), a cui spetta di conseguenza sempre l'ID 1 della tabella.

La configurazione del processo init(8) avviene attraverso il file /etc/inittab, in cui è definito il comportamento del sistema nel caso di eventi principali, come l'inizializzazione o la richiesta di reset attraverso la contemporanea pressione dei tasti [Crtl-Alt-Del], o la chiusura programmata del sistema (denominata shutdown).

In particolare, durante l'inizializzazione, viene attivata la creazione di tutto l'insieme di processi necessari al setup iniziale della macchina, il cui output diagnostico compone la lunga lista di messaggi che appaiono sullo schermo al bootstrap della macchina. Viene anche definito, per ogni terminale attivo, il comportamento che consente la connessione d'utente. A titolo d'esempio si può mostrare che la riga del file /etc/inittab:

```
5:2345:respawnrespawn:/sbin/
mingetty tty5
```

definisce il comportamento del terminale connesso con il dispositivo di console n. 5 (ovvero il dispositivo di I/O costituito da video e tastiera, direttamente connesso al personal computer, attivabile attraverso la sequenza di tasti [Alt-F5]), al quale è legato il programma mingetty(8), una versione minimale del getty(1) adatta

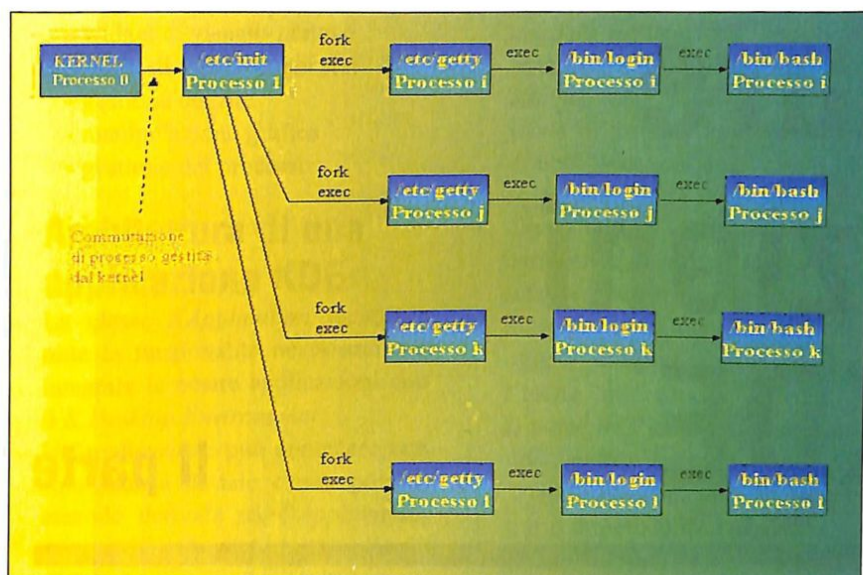



Figura 4 – Relazioni tra alcuni componenti del sistema Unix in esecuzione. Il kernel attiva attraverso una commutazione di processo il programma init(8), il quale crea tanti processi quante console definite, associando a ciascuna il comando getty(8). Nel momento in cui un utente risponde al prompt il comando login(1) autenticherà la sessione e aprirà la shell predefinita dell'utente.

ai dispositivi di console ma non alle linee seriali, che permette l'accesso dell'utente attraverso la visualizzazione del classico messaggio di login.

La Figura 4 mostra la relazione tra alcuni componenti del sistema Unix in esecuzione. Il kernel attiva attraverso una commutazione di processo, il programma init(8), il quale crea tanti processi quante sono le console definite, associando a ciascuna il comando mingetty(8). Nel momento in cui un utente risponde al prompt:

```
Red Hat Linux release 6.0 (Hedwig)
kernelkernel 2.2.5-15 on i686
```

maverick login:

il programma login(1) sostituisce il mingetty(8), e dopo le operazioni di autenticazione e di creazione dell'ambiente di lavoro, viene a sua volta sostituito dalla shell predefinita per l'utente, solitamente bash(1). C'è da notare che in questi ultimi due passaggi, non viene mai creato un nuovo processo. Solo init(8) ha creato il processo attraverso una chiamata di sistema denominata fork(2), il passaggio, invece, da un comando all'altro, avviene attraverso una diversa chiamata denominata execve(2). Al termine della sessione di lavoro dell'utente il processo viene semplicemente "ammazzato" con la chiamata di sistema kill(2). La console collegata si trova quindi orfana del proprio processo di gestione, fino a quando il kernel non comunica ad init(8) di occuparsene di nuovo. A questo punto init(8) crea un nuovo processo con la fork(2) e riesegue il programma mingetty(8); il ciclo ricomincia (questo è quanto gli viene imposto dall'opzione respawn).

Questo esempio mostra come anche un programma di utilità fondamentale per la gestione del sistema, lavori esclusivamente a livello di applicazione e non come parte integrante del sistema operativo stesso, comunicando con il kernel solo attraverso interfacce ben definite delle chiamate di sistema e dei file-speciali per i dispositivi esterni.

Conclusioni

Unix non è un sistema giovane. Aveva una base installata già molto ampia quando le prime copie di MS-DOS venivano appena prodotte. Con l'avvento del nuovo millennio, si festeggeranno per Unix i 30 anni di vita.

Un periodo nel quale altri sistemi operativi sono nati, si sono estesi dovunque e sono deceduti senza quasi lasciare altra traccia se non residui mitici. Unix si è evoluto pur rimanendo sostanzialmente coerente con il progetto originale di Ken Thompson, Brian Kernigham e Dennis Ritchie.

La linfa che ha tratto dalle distribuzioni di pubblico dominio, prima dei programmi della Free Software Foundation con il progetto GNU e successivamente con Linux, ne stanno facendo una praticabile alternativa al monopolio dei sistemi operativi proprietari come quelli di Microsoft o Apple. È un sistema operativo relativamente piccolo ed è, senza dubbio, possibile acquisirne una discreta padronanza in tempi limitati. Tutto quello che può essere ricercato attraverso il sistema di help in linea di Unix è solitamente riportato sulla documentazione tecnica, scrivendone il nome seguito da un numero tra parentesi, ad esempio vi(1). Il numero indica la sezione del manuale che contiene la pagina relativa al comando. Con il comando "$ man vi" (o "$ man 1 vi" laddove sorgessero ambiguità) è possibile avere una completa spiegazione sulla funzione utilizzata. La presenza del numero è anche un esplicito invito a consultare la relativa pagina di manuale.

Emmanuele Somma

Sviluppare applicazioni con il KDE

II parte

Le librerie fornite insieme al KDE estendono le funzionalità della libreria Qt, offrendo una serie di classi che permettono allo sviluppatore di rendere le proprie applicazioni perfettamente integrate con il desktop e il Window Manager. In questo articolo, impareremo ad utilizzare le classi della libreria, seguendo le varie fasi dello sviluppo di una completa applicazione KDE

Il *K Desktop Environment* non può essere considerato un banale Window Manager, bensì un ambiente grafico completo ed elegante, che continua ad evolversi e a maturare ad un ritmo elevatissimo, grazie all'ineccepibile lavoro del KDE team.
Tutti i servizi fondamentali, offerti da questa affascinante interfaccia grafica, possono essere integrati rapidamente nelle nostre applicazioni, utilizzando le librerie di base del KDE.
Sostanzialmente, le classi presenti nelle varie librerie possono essere raggruppate nelle seguenti categorie:

1) Classi base dell'applicazione
- finestre principali
- oggetti per interagire con il sistema KDE

2) Elementi dell'interfaccia grafica:
- finestre di dialogo
- widget di controllo (combo box, button, ecc...)
- menu

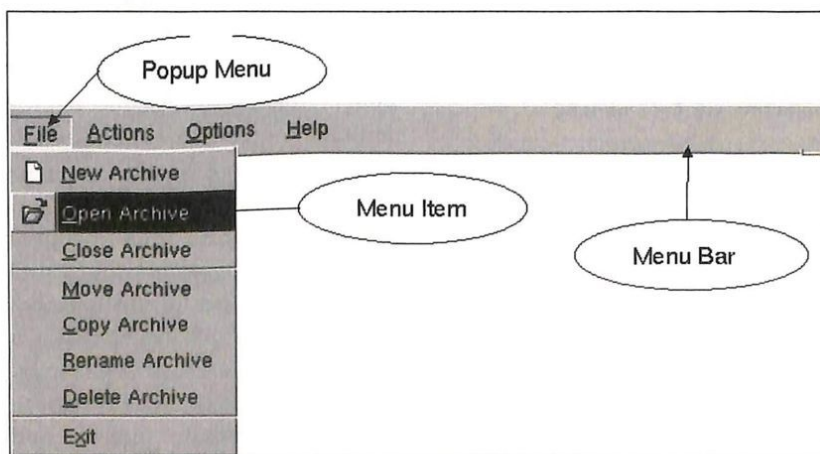



Figura 1– Struttura dei menu KDE

- widget di visualizzazione

3) Classi di utilità generale:

- gestione dei file
- manipolazione grafica
- gestione dei processi

Architettura di una applicazione KDE

La classe *KApplication* incapsula tutte le funzionalità necessarie per integrare le nostre applicazioni con il *K Desktop Environment*.

Un'applicazione può contenere una sola istanza di tale classe poiché, essendo derivata da *QApplication*, gestisce la coda degli eventi generati dal sistema grafico X Windows inviandoli a tutti i widget figli della stessa applicazione.

KApplication fornisce le seguenti funzionalità aggiuntive rispetto a *QApplication:*

- una serie di metodi per accedere al file-system del KDE;
- un oggetto per la gestione della sessione;
- il supporto multilingue tramite un oggetto di tipo *Klocale;*
- una serie di funzioni di supporto per le operazioni comuni a tutte le applicazioni KDE.

Possiamo accedere all'oggetto applicazione tramite il metodo statico *KApplication::getKApplication()* che restituisce un puntatore all'istanza di *KApplication*, oppure, utilizzando la macro kapp definita nel file *"kapp.h"* dove, tra l'altro, è dichiarata *KApplication* con tutte le sue classi di supporto.

La funzione *kapp->getConfig()* restituisce un oggetto di tipo *KConfig* che permette di leggere e scrivere dei valori di configurazione su un file di risorse standard associato all'applicazione.

Mediante l'istanza predefinita di *KIconLoader* possiamo caricare le icone in formato xpm negli oggetti di tipo *QPixmap*.

In genere, non è necessario scomodare i metodi di *KIconLoader* per caricare le pixmap; è sufficiente utilizzare la macro *Icon ("myicon.xpm")*, la quale restituisce un oggetto di tipo *Qpixmap* contenente l'icona *"myicon.xpm"*.

Il nome dell'icona può essere anche un path assoluto del tipo *"/mydir/myicon.xpm"*, in caso contrario il metodo *KIconLoader::loadIcon()*, invocato dalla macro, cercherà il file specificato nelle directory indicate nel path di ricerca standard:

```
1: $HOME/.kde/share/apps/pics
2: $KDEDIR/share/apps/pics
3: $HOME/.kde/share/apps/toolbar
4: $KDEDIR/share/apps/toolbar
5: $HOME/.kde/share/icons
6: $HOME/.kde/share/toolbar
7: $KDEDIR/share/icons
8: $KDEDIR/share/toolbar
```

Se necessario, mediante la funzione *KIconLoader::insertDirectory()* è possibile aggiungere una nuova directory al path di ricerca.

Il supporto per l'internazionalizzazione delle risorse del sistema viene gestito dall'oggetto *KLocale* in tandem con *KCharsets* il quale rende disponibili le informazioni sul set di caratteri corrente utilizzato dal KDE.

Le funzioni per accedere a tali oggetti sono rispettivamente *KApplication::getLocale()* e *KApplication::get Charsets()*.

Infine possiamo intercettare i signal *saveYourself()* e *shutDown()*, emessi da *KApplication* quando viene chiusa la sessione KDE, per implementare le funzioni di salvataggio dei dati dell'applicazione.

La finestra principale KTMainWindow

Ogni volta che creiamo un oggetto, derivato da *QWidget*, dobbiamo specificare nel costruttore almeno il puntatore al widget di appartenenza. I widget principali, chiamati anche *'top-level widget'* non appartengono, ovviamente, a nessun altro widget e sono gestiti direttamente dall'istanza di *KApplication*.

Per essere visibile sul desktop, un'applicazione necessita di un widget principale che può essere

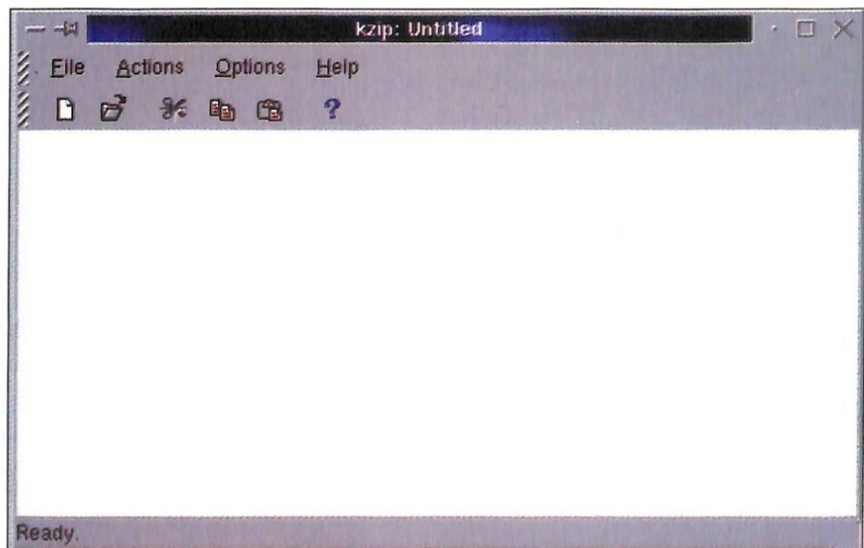



Figura 2 – Finestra di KZip dopo le modifiche

un semplice pulsante, oppure un'istanza di *KTMainWindow* (una complessa finestra principale pronta per l'uso, completa di menu bar, status bar, tool bar e funzioni per la gestione della sessione).

In genere, è consigliabile impostare il widget principale tramite il metodo *Kapplication::setTopWidget()*, dato che esso provvede automaticamente a connettere lo slot *Qapplication::quit()* con il signal generato alla chiusura della finestra principale.

Se utilizziamo *KTMainWindow* come widget principale, il metodo *setTopWidget()* verrà invocato direttamente nel costruttore della classe.

Un'applicazione contenente più top-level widget può essere comunque terminata connettendo il signal *Qapplication::lastWindowClosed()*, inviato dall'ultima finestra chiusa, con lo slot *Qapplication::quit()* nel seguente modo:

```
int main(int argc, char* argv[])
{
  KApplication kapp(argc,argv,"
                          myfirstapp");
  QObject::connect(kapp, SIGNAL
          (lastWindowClosed()), kapp,
                        SLOT(quit()));
  MainWindow* main=new
          MainWindow("myfirstapp");
  main->show();
  return kapp.exec();
}
```

Riassumendo, i passaggi fondamentali per inizializzare una applicazione KDE sono:

a) creazione dell'istanza di *KApplication* all'inizio della funzione *main()*;
b) connessione dello slot *QApplication::quit()* con l'evento generato dalla chiusura della finestra principale;
c) creazione ed impostazione del top-level widget;
d) esecuzione dell'applicazione mediante il metodo *Qapplication::exec()*.

Creazione dello scheletro dell'applicazione

Ora inizieremo a costruire un'applicazione completa, cercando di sfruttare, nel migliore dei modi, i servizi offerti dal KDE.

Utilizzeremo come ambiente di sviluppo il Kdevelop 1.0 Beta3 che troverete incluso nel cd-rom allegato alla rivista, insieme ai sorgenti dell'applicazione (cartella *"kzip-project"*), e della recente versione 1.1.2 del KDE.

Uno dei programmi più utilizzati dagli utenti Windows è certamente il famoso WinZip, tant'è vero che sembra ormai impossibile trovare installazioni del sistema operativo di Zio Bill, prive di questo utilissimo tool.

Tra i programmi forniti con il KDE esiste un tool simile che permette di gestire gli archivi zip e tgz ma, a differenza del WinZip, esso presenta un interfaccia poco intuitiva e povera di opzioni.

La nostra applicazione, che chiameremo con molta originalità *"KZip"*, cercherà di colmare le lacune dell'utilità di archiviazione di sistema, offrendo all'utente un'interfaccia più completa e funzionale.

Poiché il nostro *"KZip"* avrà una semplice interfaccia con vista unica, e la solita dotazione di tool bar, menu bar e status bar, possiamo risparmiare un po' di linee di codice generando lo scheletro del programma mediante il wizard del *Kdevelop*.

Dal menu *"progetto->nuovo"* lanciate l'*Application Wizard* e selezionate il tipo di applicazione *"KDE normale"*.

Passate al prossimo step e inserite come nome del progetto *"KZip"*. Riempite le rimanenti caselle di testo nel modo opportuno e procedete fino alla creazione del progetto.

Notate che il wizard ha creato per noi, oltre alla classe principale *KZipApp*, due classi accessorie: *KZipDoc* e *KZipView*.

Queste classi servono a supportare il modello *document-view* utilizzato spesso nei word processor ma, poiché la nostra applicazione avrà una singola view per ogni documento (in questo caso un file archivio), per il momento possiamo tranquillamente ignorarle.

I menu *"view"* e *"edit"*, indubbiamente non hanno nulla da spartire con il nostro *"KZip"*, per cui provvederemo a rinominarli clonando spudoratamente la barra menu di WinZip!

Prima di proseguire, è necessario comprendere il meccanismo usato dalla classe *KTMainWindow* per gestire menu e toolbar.

La barra dei menu

Gli elementi fondamentali per gestire i menu sono:

- La barra menu rappresentata dal componente *KMenuBar*, ovvero un semplice frame widget che contiene i menu;
- i menu contenenti i menu item, realizzati mediante istanze della classe *QPopupMenu*.

Esiste un solo oggetto di tipo *KMenuBar* nella *KTMainWindow*, ed è accessibile tramite la funzione *KTMainWindow::menuBar()*.

La posizione e il ridimensionamento della barra menu sono gestiti automaticamente dalla finestra principale; di conseguenza può essere spostata in alto e in basso oppure trascinata al di fuori dell'area visibile.

Nella classe *KzipApp,* la barra menu viene creata e riempita all'interno della funzione *KZipApp::init Menubar()* nel modo seguente:

```
...
// menuBar entry edit_menu
edit_menu = new QPopupMenu();
edit_menu->insertItem(Icon ("edit
                cut.xpm"), i18n("Cu&t"),
                          ID_EDIT_CUT );
edit_menu->insertItem(Icon
        ("editcopy.xpm"), i18n("&Copy"),
                          ID_EDIT_COPY );
edit_menu->insertItem(Icon
        ("editpaste.xpm"), i18n("&Paste"),
                          ID_EDIT_PASTE );

//////////////////////////////////////////
// menuBar entry view_menu
view_menu = new QPopupMenu();
view_menu->setCheckable(true);
view_menu-
>insertItem(i18n("&Toolbar"),
ID_VIEW_TOOLBAR);
view_menu->insertItem(i18n
("&Statusbar"), ID_VIEW_STATUSBAR );
...
```

Tutte le variabili di tipo *QPopupMenu** sono dichiarate nella sezione private della classe *KZipApp.*

La creazione di un menu è un operazione piuttosto semplice: è sufficiente creare l'oggetto *QPopupMenu* e inserirvi i vari menu item utilizzando il metodo *QPopupMenu::insertItem (const QPixmap & pixmap, const char * text, int id= -1, int index=-).*

I parametri della funzione rappresentano nell'ordine:

- l'icona che comparirà alla sinistra del menu *item,* l'etichetta del menu *item*, l'ID che identifica il menu item e la posizione in cui inserire l'elemento;
- un menu *item* può essere una stringa di testo, un icona, un separatore oppure un oggetto *QPopupMenu,* e l'inserimento dei vari tipi di dati è gestito in modo trasparente dal metodo *insertItem()* grazie alla tecnica dell'overloading.

Osservate come sia semplice e naturale aggiungere al menu *"File"* il sub-menu *"Open Recent"*:

```
recent_files_menu = new
QPopupMenu();
...

//////////////////////////////////////
// menuBar entry file_menu
file_menu = new QPopupMenu();
...
file_menu->insertSeparator();
file_menu->insertItem(Icon
        ("filenew.xpm"), i18n("&New"),
                          ID_FILE_NEW );
...
file_menu->insertItem(i18n("Open
        &recent"), recent_files_menu,
                   ID_FILE_OPEN_RECENT );
```

Alla fine tutti i menu creati vengono inseriti nella barra menu:

```
menuBar()->insertItem(i18n("&File"),
                              file_menu);
menuBar()->insertItem(i18n("&Edit"),
                              edit_menu);
...
```

Toolbar

E' possibile aggiungere alla finestra principale, un numero arbitrario di toolbar mediante la funzione *KTMainWindow::toolBar (int id).*

Una chiamata a *toolBar(id)* crea, se non esiste già, un nuovo oggetto di tipo *KToolBar* identificato dal valore *'id'* e ne restituisce il puntatore. La classe *KToolBar* fornisce numerosi metodi per aggiungere alla toolbar pulsanti, combo box, separatori e widget di ogni tipo. Come al solito è sufficiente dare uno sguardo al codice sorgente per eliminare eventuali dubbi e perplessità:

```
void KZipApp::initToolBar()
{

//////////////////////////////////////
// TOOLBAR
toolBar()
->insertButton(Icon("filenew.xpm"),
ID_FILE_NEW, true, i18n("New File") );
toolBar()
->insertButton(Icon("fileopen.xpm"),
ID_FILE_OPEN, true, i18n("Open File")
                                      );
toolBar()
->insertButton(Icon ("filefloppy.xpm"),
 ID_FILE_SAVE, true, i18n("Save File")
);
toolBar()
->insertButton(Icon("fileprint.xpm"),
  ID_FILE_PRINT, true, i18n("Print") );
toolBar()->insertSeparator();
...
}
```

La funzione:

```
KToolBar::insertButton(const QPixmap &
    pixmap, int ID, bool enabled = true,
       const char *ToolTipText = 0L, int
                               index=-1)
```

aggiunge alla toolbar un pulsante completo di icona e tooltip. I parametri, in questo caso, identificano rispettivamente l'icona da associare al button, l'id, un valore booleano che stabilisce se attivarlo o meno, e infine, la stringa di testo del tooltip. Le toolbar, come la barra dei menu, possono fluttuare liberamente e in più sono capaci di attaccarsi a qualunque lato della finestra.

Naturalmente, se desiderate conoscere tutti i dettagli delle classi *KToolBar* e *KMenuBar,* consultate la guida in linea del *Kdevelop.*

L'interfaccia di KZip

Sostanzialmente, gli elementi dell'interfaccia grafica della nostra applicazione dovranno essere mo-

Menu "File"

- **"New Archive"**
- **"Open Archive"**
- **"Close Archive"**
- **"Move Archive"**
- **"Copy Archive"**
- **"Delete Archive"**
- **"Rename Archive"**
- **"Exit"**

Menu "Actions"

- "Add"
- "Delete"
- "Extract"
- "View"
- "Select All"

Menu "Options"

- "Configuration"
- "Save settings on exit"
- "Save settings now"

Pulsanti Toolbar

- "New"
- "Open"
- "Add"
- "Extract"
- "View"

dificati nel seguente modo:

Dopo aver modificato opportunamente le funzioni *initMenubar()* e *initToolbar()*, occorre definire, nel file *"resource.h"*, gli identificatori da associare ai menu item e ai pulsanti della toolbar. In Figura 2, potete osservare l'aspetto dell'applicazione dopo il "restyling".

Gestione degli eventi

Ora che abbiamo definito l'aspetto dell'interfaccia grafica, vediamo quanto sia semplice associare gli eventi, generati da menu item e pulsanti, alle opportune funzioni di risposta. Alla fine della funzione *initMenubar()* troviamo le seguenti istruzioni:

```
...
////////////////////////////////////////
// CONNECT THE MENU SLOTS WITH
SIGNALS
// for execution slots and statusbar
//messages
connect(file_menu,SIGNAL(activated
    (int)),SLOT(commandCallback(int)));
connect(file_menu,SIGNAL(highlighted
        (int)),SLOT(statusCallback(int)));
connect(actions_menu,SIGNAL(activa-
ted
    (int)),SLOT(commandCallback(int)));
connect(actions_menu,SIGNAL(highligh-
ted
        (int)),SLOT(statusCallback(int)));

connect(options_menu,SIGNAL(activa-
ted
    (int)),SLOT(commandCallback(int)));
connect(options_menu,SIGNAL(highli-
ghted
        (int)),SLOT(statusCallback(int)));
...
```

Un oggetto di tipo *QPopupMenu* emette il signal *activated(int id)* quando un menu item viene selezionato, e il signal *highlighted(int id)* se il menu item è semplicemente evidenziato. Lo slot *commandCallback()* ha la seguente struttura:

```
void KZipApp::commandCallback(int
id_){
  switch (id_){
      ...
      case ID_FILE_NEW:
      slotFileNew();
  break;
  case ID_FILE_OPEN:
      slotFileOpen();
  break;
  ...
  default:
  break;
}
```

Praticamente, quando selezioniamo la voce di un menu, in risposta al signal *clicked(int id)*, viene eseguito lo slot *commandCallback (int id_)*, che riceve il valore dell'id corrispondente alla voce selezionata, e passa il controllo alla funzione appropriata mediante il blocco *"switch..case"*.

Lo slot *statusCallback(int)* si occupa di visualizzare, sulla barra di stato, una breve descrizione della azione associata alla voce del menu evidenziato. I due slot *statusCallback(int id_)* e *commandCallback(int id_)* sono utilizzati nel medesimo modo all'interno di *initToolbar()* per intercettare i signal *clicked(int id)* e *pressed(int id)* generati dall'oggetto toolBar():

```
////////////////////////////////////////
// CONNECT THE TOOLBAR SLOTS
WITH
// SIGNALS - add new created toolbars
// by their according number
// connect for invoking the slot actions
connect(toolBar(), SIGNAL(clicked(int)),
        SLOT(commandCallback(int)));
// connect for the status help on holing
// icons pressed with the mouse button
connect(toolBar(),
SIGNAL(pressed(int)),
        SLOT(statusCallback(int)));
```

Grazie a questo tipo di meccanismo, i passi da compiere per aggiungere eventualmente nuovi tool button e/o menu item all'interfaccia grafica, sono semplici e immediati; dopo aver inserito i nuovi elementi nella *toolbar/ menubar* è sufficiente aggiungere all'interno dello slot *commandCalback()* (ed eventualmente in *statusCalback()*), queste poche righe:

```
switch(id_) {
  ...
  case ID_NUOVO_ITEM:
      slotNuovoItem();
      break;
      ...
}
```

dove *ID_NUOVO_ITEM* è l'id assegnato al nuovo elemento e *slotNuovoItem()* è la funzione a cui passare il controllo dell'applicazione.

Conclusioni

Bene.... a questo punto lo spazio a nostra disposizione è esaurito...!

Nel prossimo numero ci occuperemo della progettazione dei componenti necessari alla visualizzazione degli archivi compressi, e implementeremo una semplice interfaccia per interagire con i tool di compressione/decompressione più diffusi (tar, bzip, gzip, zip, ecc...).

Stefano Frangella

Web Scripting PHP3 e i database

Le funzioni per l'accesso ai database relazionali sono il vero punto di forza di PHP3. Nel linguaggio se ne possono trovare davvero molte. In pratica vengono soddisfatte tutte le esigenze di interfacciamento con le basi di dati più diffuse

Se vi ricordate, già nel primo articolo avevamo presentato la numerosa lista dei database supportati da PHP3. Si trattava di un elenco veramente completo. Accanto a marchi prestigiosi, quali Oracle e Sybase, c'erano nomi ben più conosciuti al popolo dei linuxiani come mSQL, MySQL e PostgreSQL. Non poteva poi mancare la sigla ODBC, la cui presenza faceva subito intuire che con PHP saremmo stati in grado di usare gli archivi attraverso una fonte di dati ODBC. Effettivamente, il programmatore può avere problemi di scelta di fronte al numero di driver offerti da PHP. Per non rimanere intrappolati dallo stesso problema, concentriamoci sulle funzioni per l'accesso ai database più usati nei server Linux. Attenzione al fatto che per testare con successo gli esempi presentati in questo articolo, si dovranno "passare" i seguenti parametri al comando "configure" eseguito all'inizio della fase di compilazione del pacchetto PHP:

```
—with-mysql=<DIR>
—with-pgsql=<DIR>
—with-msql=<DIR>
```

Dove DIR è da sostituire con il path della directory di installazione del db.

PHP3 E GDBM

Il modo più semplice per archiviare dati in una macchina Linux consiste nel creare e riempire un database DBM. In realtà il termine DBM non indica il formato dell'archivio bensì una libreria di funzioni per scrivere, cancellare, ricercare o sostituire coppie di dati (chiave/valore) in un file. Si tratta, inoltre, di una libreria standard degli ambienti Unix e può essere facilmente inclusa nei programmi durante la fase di linking. Ovviamente, anche Linux possiede una libreria simile. Il suo nome è GDBM, in virtù della filosofia GNU (GNU is Not Unix) che sta alla base di ogni progetto per Linux. È quindi un free software, infatti viene distribuito rispettando i principi della licenza GPL (General Public License), ma anche totalmente compatibile con la classica DBM. È bene precisare che esistono alcune varianti alla libreria DBM. Le più note sono Berkeley-DB e NDBM, entrambe supportate da PHP3. Un argomento interessante su cui bisogna fare attenzione, quando si lavora con file ed archivi, è il criterio di locking usato da PHP per bloccare in modo esclusivo l'accesso in scrittura ai database DBM. In realtà, è la libreria stessa che si incarica del blocco dell'archivio. PHP non fa altro che effettuare un lock secondario, tra l'altro sempre eseguito automaticamente quando si scrive su un file, senza interferire con il blocco primario.

Dopo questa breve introduzione teori-

Link

http://www.sleepycat.com
Per documentarsi bene sulla libreria Berkeley-DB

http://www.postgresql.org
Per scaricare la versione corrente di PostgresSQL

http://www.mysql.com
Per chi volesse sapere tutto su MySQL

http://www.Hughes.com.au
Per avere la versione aggiornata di mSQL

ca, passiamo alla pratica, scoprendo il modo in cui si accede ad un archivio in stile DBM usando la libreria GDBM.
La funzione principale, usata per leggere o creare l'archivio, è

```
int dbmopen(string filename, int flags);
```

Il primo parametro è il nome del file dell'archivio, eventualmente completo di path. Il secondo, invece, è il modo in cui si accede al file. L'argomento "flags", quindi, può avere i seguenti valori:

'r' Apre l'archivio in sola lettura;
'n' Apre l'archivio in lettura e scrittura e tronca il file se esiste già;
'w' Apre l'archivio in lettura e scrittura;

Dopo la chiamata a dbmopen(), che restituisce una sorta di puntatore al file del db, saremo in grado di compiere le operazioni di inserimento, ricerca, sostituzione o cancellazione delle coppie di dati utilizzando nell'ordine le seguenti funzioni.

```
int dbminsert(int dbm_identifier, string
                         key, string value);
string dbmfetch(int dbm_identifier, string key);
bool dbmreplace(int dbm_identifier,
                    string key, string value);
bool dbmdelete(int dbm_identifier, string key);
```

Vediamo un'applicazione pratica, proponendo un esempio per gestire un semplice archivio di login e password cifrate. Iniziamo dal form html

```
<html> <head>
<title>Esempio GDBM</title>
</head> <body>
<form method=post
    action=http://localhost/gdbm.php3>
Login: <input type=text size=8
                        name=login><p>
Password: <input type=text size=8
                     name=password><p>
<input type=radio name=azione
                   value=CREA>Crea<br>
<input type=radio name=azione
                 value=CERCA>Cerca<br>
<input type=radio name=azione
             value=ELIMINA>Elimina<br>
<input type=radio name=azione
```

```
           value=MODIFICA>Modifica<br>
<input type=radio name=azione
                 value=ELENCA>Elenca<p>
<input type=submit value=Ok>
<input type=reset value=Annulla>
</form> </body> </html>
```

Figura 1: Esempio GDBM; inserimento delle password.

e poi terminiamo il lavoro con il programma PHP:

```
<html> <head>
<title>Esempio GDBM</title>
</head> <body>
<?php
  $pwd = crypt($password, "pw");
  $dbm = dbmopen(
          "/www/dati/users.dbm", "w");
  switch($azione) {
  case "CREA" :
    if(dbmexists($dbm, $login)) {
      echo "Attenzione!!!<br>";
      echo "User già presente nell'archivio";
    }
    else {
      dbminsert($dbm, $login, $pwd);
      echo "Dati creati con successo";
    }
    break;
  case "CERCA" :
    $key = dbmfetch($dbm, $login);
    if($key) {
      echo "$login -> $key<br>";
    }
    else {
      echo "Dato non trovato";
    }
    break;
  case "ELIMINA" :
    if(! dbmdelete($dbm, $login)) {
```

```
      echo "Dato eliminato con successo";
    }
    else {
      echo "Impossibile eliminare lo user
$login";
    }
    break;
  case "MODIFICA" :
    if(! dbmreplace($dbm, $login, $pwd)) {
      echo "Dato modificato con successo";
    }
    else {
      echo "Impossibile modificare i dati";
    }
    break;
  case "ELENCA" :
    echo "<table border>";
    echo "<tr><td>Login</td><td>
                     Password</td></tr>";
    // ---
    $key = dbmfirstkey($dbm);
    while($key) {
      echo "<tr>";
      echo "<td>$key</td><td>" .
        dbmfetch($dbm, $key) . "</td>";
      echo "</tr>";
      $key = dbmnextkey($dbm, $key);
    }
    // ---
    echo "</table>";
    break;
  }
  dbmclose($dbm);
?>
</body> </html>
```

Figura 2: Esempio GDBM: Elenco delle password.

Forse, per pura pignoleria, l'ultima parte dell'esempio non ci convincerà più di tanto. L'elenco delle coppie login/password, infatti, non è ordinato. Per rimediare potremo creare un array di appoggio ed ordinarlo con la funzione asort(). Vediamo:

```
$key = dbmfirstkey($dbm);
while($key) {
  $users[] = $key;
  $key = dbmnextkey($dbm, $key);
}
sort($users);
$i = sizeof($users);
for($k=0; $k<=$i; $k++) {
  echo "<tr><td>$users[$k]</td> <td>"
. dbmfetch($dbm, $users[$k]) . "</td></tr>";
}
```

Dopo aver sostituito le righe di codice racchiuse dai commenti //— con quest'ultime, dovremo ottenere la lista degli user ordinata per login.

PHP3 e POSTGRESQL

Se le nostre esigenze di archiviazione dovessero aumentare, e quindi ci fosse bisogno di accedere a basi di dati ben più complesse dei file GDBM, dovremmo usare il linguaggio SQL adottando un vero gestore di database relazionali (RDBMS).

Anche in questo caso non ci dovremo preoccupare più di tanto, poiché PHP3 e Linux ci daranno una grossa mano per risolvere brillantemente ogni nostro problema.

Supponiamo, quindi, di dover gestire un piccolo negozio di componenti hardware per personal computer. Semplificando molto il problema e tralasciando le nozioni teoriche relative alla creazione ed al mantenimento delle basi di dati relazionali, concentreremo la nostra attenzione sulle principali funzioni di PHP per accedere ad un database di nome "negozio" creato con PostgreSQL, e costituito dalle seguenti tabelle:

Tabella Prodotti

cprod	codice del prodotto
cat	categoria del prodotto
descp	descrizione del prodotto
prezzo	prezzo unitario
qres	quantità residua

Tabella Clienti

ccli	codice del cliente
descc	descrizione del cliente
ind	indirizzo
piva	partita iva
tel	telefono
fax	fax
email	indirizzo email

Tabella Ordini

cord	codice dell'ordine
ccli	codice del cliente
cprod	codice del prodotto
qord	quantità ordinata
data	data dell'ordine

Una delle operazioni, che dovremo essere in grado di eseguire con il nostro browser preferito, sarà l'inserimento o la modifica dei dati relativi ad un prodotto venduto dal negozio. Prepariamo subito il form html di appoggio:

```
<html> <head>
<title>Nuovo prodotto</title>
</head><body>
<form method=post action=http://localhost/
                        nuovoprodotto.php3>
Codice Prodotto: <input type=text size =
                    4 name=cprod><p>
<select name=cat>
<option value=CPU>CPU
<option value="Schede video">Schede video
<option value="Hard disk">Hard disk
</select><p>
Descrizione: <input type=text size=50
                    name=descp><p>
Prezzo: <input type=text size=5
                    name=prezzo><p>
Quantita' residua: <input type=text
                size=3 name=qres><p>
<input type=submit value=Ok>
<input type=reset value=Annulla>
</form> </body> </html>
```

Poi ci servirà lo script "nuovoprodotto.php3" che alimenterà la tabella prodotti con i dati ricevuti dal form, eccolo:

```
<?php
  $db = pg_connect("localhost", "5432",
                    "", "", "negozio");
  if(!$db) {
    echo "<h1>Connessione al database
            "NEGOZIO" fallita</h1>";
    exit;
  }
  pg_exec($db, "INSERT INTO prodotti VALUES
      ($cprod, $cat, $descp, $prezzo, $qres)");
  pg_close($db);
?>
```

Potrebbe servirci anche il listino completo dei prodotti in catalogo. In questo caso dovremo scrivere un codice simile al seguente:

```
<table border>
<tr>
<td>Codice prodotto</td>
<td>Categoria</td>
<td>Descrizione</td>
<td>Prezzo</td>
</tr>
<?php
  $query = pg_exec($db, "SELECT * FROM
      prodotti ORDER BY cat AND descp");
  $row = 0;
  while ($data = pg_fetch_object
                    ($query, $row)) {
    echo "<tr><td>" . $data->cprod .
                    "</td></tr>";
    echo "<tr><td>" . $data->cat .
                    "</td></tr>";
    echo "<tr><td>" . $data->descp .
                    "</td></tr>";
    echo "<tr><td>" . $data->prezzo .
                    "</td></tr>";
    $row++;
  }
?> </table>
```

PHP3 E MySQL

Immaginiamo adesso di dover eseguire le precedenti operazioni sulla tabella clienti, ipotizzando però, che il database del negozio sia interamente gestito dall'RDBMS MySQL.

Quest'ultimo, grazie alle sue doti di velocità e stabilità, rappresenta una validissima alternativa al PostgreSQL. Iniziamo dal modulo per inserire i dati dei clienti:

```
<html> <head>
<title>Nuovo cliente</title>
</head> <body>
<form method=post action=http://localhost
                    /nuovocliente.php3>
Codice cliente: <input type=text size=4
```

```
                                name=ccli><p>
Descrizione: <input type=text size=50
                              name=descc><p>
Indirizzo: <input type=text size=50
                                name=ind><p>
Partita IVA: <input type=text size=15
                               name=piva><p>
Telefono: <input type=text size=15
                                name=tel><p>
Fax: <input type=text size=15
                                name=fax><p>
E-mail: <input type=text size=20
                              name=email><p>
<input type=submit value=Ok>
<input type=reset value=Annulla>
</form> </body> </html>
```

e proseguiamo con il frammento di programma PHP contenente le funzioni di accesso al db MySQL:

```
<?php
 $dblink = mysql_connect("localhost");
 if(!$dbconn) {
   echo "<h1>Connessione al database
                          fallita</h1>";
   exit;
 }
 $db = mysql_select_db("negozio", $dblink);
 if(!$db) {
   echo "<h1>Impossibile aprire il
               database "NEGOZIO"</h1>";
   exit;
 }
 mysql_query("INSERT INTO clienti
      VALUES ($ccli, $descc, $ind, $piva,
                    $tel, $fax, $email)");
 mysql_close($db);
?>
```

se volessimo ottenere un elenco di tutti i clienti del negozio; altrimenti dovremmo usare questo script:

```
<table border>
<tr>
<td>Codice cliente</td>
<td>Descrizione</td>
<td>Indirizzo</td>
<td>Partita IVA</td>
<td>Telefono</td>
<td>Fax</td>
<td>E-mail</td>
</tr>
<?php
 $sx_cell = "<tr><td>";
 $dx_cell = "</td></tr>";
 $query = mysql_db_query("db",
"SELECT * FROM clienti");
 while($row=mysql_fetch_array($query)) {
   echo $sx_cell $row["ccli"] $dx_cell;
   echo $sx_cell $row["descc"] $dx_cell;
   echo $sx_cell $row["ind"] $dx_cell;
   echo $sx_cell $row["piva"] $dx_cell;
   echo $sx_cell $row["tel"] $dx_cell;
   echo $sx_cell $row["fax"] $dx_cell;
   echo $sx_cell $row["email"] $dx_cell;
 }
 mysql_free_result($result);
?> </table>
```

PHP3 E mSQL

Concludiamo presentando alcune funzioni per accedere ad un database costruito con mSQL. Anche questo RDBMS è molto conosciuto tra gli utilizzatori di Linux. Questa volta useremo la tabella ordini per testare gli esempi. Iniziamo dal consueto form per l'inserimento dei dati:

```
<html> <head>
<title>Nuovo ordine</title>
</head> <body>
<form method=post action=http://localhost
                         /nuovoordine.php3>
Codice ordine: <input type=text size=4
                           name=cord><p>
Codice cliente: <input type=text size=4
                           name=ccli><p>
Codice prodotto: <input type=text
                   size=4 name=cprod><p>
Data: <input type=text size=10
                           name=data><p>
Quantita': <input type=text size=2
                           name=qord><p>
<input type=submit value=Ok>
<input type=reset value=Annulla>
</form> </body> </html>
accompagnato dal codice PHP3
<?php
 $db = msql_connect("");
 if(!$db) {
   echo "<h1>Connessione al database
                         fallita</h1>";
   exit;
 }
 msql_query("INSERT INTO ordini VALUES
   ($cord, $ccli, $cprod, $data, $qord)");
 msql_close($db);
?>
```

Per ottenere la stampa di un elenco degli ordini di un particolare cliente, ordinato per data di acquisto, si potrebbe usare la funzione msql_fetch_row() come nell'esempio seguente:

```
<table border>
<tr>
<td>Codice ordine</td>
<td>Codice prodotto</td>
<td>Prodotto</td>
<td>Quantità ordinata</td>
<td>Data dell'ordine</td>
</tr>
<?php
 $sx_cell = "<tr><td>";
 $dx_cell = "</td></tr>";
 $query = msql_query("SELECT cord.ordini,
   cprod.prodotti, descp.prodotti, qord.ordini,
data.ordini FROM ordini, prodotti, clienti WHERE
$ccli=ccli.clienti" ORDER BY data.ordini, $db);
 while($campo = mysql_fetch_row($query)) {
   echo $sx_cell $campo[0] $dx_cell;
   echo $sx_cell $campo[1] $dx_cell;
   echo $sx_cell $campo[2] $dx_cell;
   echo $sx_cell $campo[3] $dx_cell;
   echo $sx_cell $campo[4] $dx_cell;
 }
?> </table>
```

Tutti gli esempi dimostrano come le funzioni per operare con i database più diffusi siano molto simili.
Sostanzialmente tutte restituiscono le stesse strutture dati (oggetti, array o variabili scalari) e tutte richiedono gli stessi parametri.
Cambia solo una parte del nome della funzione e l'ordine con cui vengono passati gli argomenti.
Non ci rimane altro da dire che PHP3 si conferma ancora una volta un ottimo concorrente alla tecnologia ASP, soprattutto nell'attività attualmente più impegnativa: la gestione delle basi di dati relazionali.

SU INTERNET

Per documentarsi bene sulla libreria Berkeley-DB: *http://www.sleepycat.com.* Per scaricare la versione corrente di PostgresSQL collegarsi a *http://wwwpostgresql.org*. Per chi volesse sapere tutto su MySQL consigliamo di approdare su *http://www. mysql.com.* Per avere la versione aggiornata di mSQL seguire questo link: *http:// www.Hughes.com.au.*

Francesco Munaretto

RPM, un sistema di gestione per pacchetti software

All'atto della prima installazione Linux sul proprio PC o sul server, l'utente ha a disposizione una vasta serie di moduli e applicazioni per tutte le esigenze, dagli editor di testi ultra configurabili fino ai Firewall per la sicurezza della rete. Anche se, di solito, l'utente medio utilizza solo una percentuale dell'enorme massa di applicativi e librerie installati sul sistema Linux, non sempre il software installato contiene i programmi necessari a soddisfare tutte le sue esigenze.

"Uno strumento potente e flessibile che permette di risolvere facilmente i problemi delle installazioni, senza crearne altri, e che aggiunge trasparenza sulla posizione dei file e delle librerie utilizzate."

Quando si presenterà la necessità di installare un nuovo programma, ci si imbatterà quasi sicuramente, qualunque sia la fonte di provenienza (Internet, CD-ROM acquistati in libreria o in edicola, ecc...), in un file di ampie dimensioni con una estensione particolare: la stringa *"rpm"*.

A questo punto, non rimane che rimboccarsi le maniche e capire tutto su un nuovo sistema di gestione dei pacchetti chiamato RPM, e di come sia utilissimo per migliorare il nostro nuovo sistema Linux.

Red Hat e i pacchetti RPM

La distribuzione Red Hat di Linux, ha adottato un sistema di gestione dei pacchetti software molto flessibile e potente denominato RPM. La sigla RPM è l'acronimo di **Red Hat Package Manager** ma, l'inserimento del nome di una delle più famose distribuzioni di Linux non deve sconcertare, poiché il sistema di gestione è inteso completamente come un sistema di packaging aperto e utilizzabile da tutti. Non sussistono mire di monopolio, anche se, logicamente la Red Hat cerca di diffonderlo il più possibile per standardizzare la sua distribuzione del software sotto licenza GPL (*vedi articolo sul Free Software e la Free Software Foundation*).

Anche altre distribuzioni Linux utilizzano RPM per il rilascio del software ed insieme alla Red Hat troviamo anche Caldera e SuSE.

Le potenzialità dell'RPM lo rendono uno dei sistemi di gestione più flessibili e facili da usare. Le caratteristiche più interessanti di questo gestore sono varie e ne prendiamo in considerazione solo alcune tra le più importanti.

RPM incorpora delle routine di post-installazione, che permettono di effettuare delle operazioni sui file da installare in forma compressa e apportare delle

modifiche al sistema.
E' stato progettato per effettuare delle interrogazioni molto potenti sui pacchetti e sui file che li compongono. Si possono effettuare query su tutti i pacchetti o su singoli file. Il sistema mantiene un piccolo repository, che tiene traccia delle installazioni, e di alcuni file di configurazione.
Si possono avere informazioni puntuali del singolo file trovando il pacchetto di provenienza e la sua posizione. Il file con estensione *rpm*, che contiene tutto il pacchetto, è esso stesso un file compresso, ma l'adozione di un particolare header binario permette di sapere tutto sul contenuto del pacchetto in maniera molto rapida.
Un'altra caratteristica, molto importante, è la possibilità di effettuare verifiche sui pacchetti. Ad esempio, se si è cancellato un file e si ha paura di aver rovinato un'installazione di pacchetto, basta verificarlo ed ogni anomalia verrà scoperta.
L'unico requisito che chiede l'RPM per funzionare correttamente, è che il comando *cpio* sia nella versione 2.4.2 o superiore. Questo perché il sistema di gestione dei pacchetti della Red Hat utilizza pesantemente questo comando.
Tale comando rende l'RPM anche un sistema portabile su altri sistemi operativi Unix, ma, al tempo stesso, si deve fare attenzione ad utilizzare, su una piattaforma, i pacchetti creati su un'altra piattaforma perché potrebbero non funzionare.

Breve panoramica sui comandi dell'RPM

Il programma ha una vastità di opzioni e parametri che potrebbero sconcertare l'utente alle prime armi. Digitando il comando *rpm —help,* si può vedere tutto l'elenco dei parametri dell'RPM, ma quelli di utilizzo quotidiano sono molti di meno. Di seguito viene elencata la serie di comandi più utili che servono per interrogare, installare, verificare e risolvere anomalie con i pacchetti.

Comandi per le Query

- **rpm -qpi <pacchett-rpm>**
  Mostra una descrizione del contenuto del file RPM.
- **rpm -qpl <pacchett-rpm>**
  Mostra l'elenco dei file contenuti nel file RPM e dove andranno collocati se sarà installato.
- **rpm -qa**
  Mostra l'elenco dei pacchetti RPM installati, così come sono stati registrati nel database del sistema RPM. Il repository sta sotto la directory /var/bin/rpm e si può ricreare con il comando rpm –initdb.
- **rpm –qf <file>**
  Determina il nome del pacchetto da cui proviene il file indicato come argomento.

Comandi per l'installazione

- **rpm –i <pacchetto-rpm>**
  Installa il pacchetto se non si verificano errori.
- **rpm –i —test <pacchetto-rpm>**
  Simula l'installazione di un pacchetto.
- **rpm –i —force <pacchetto-rpm>**
  Forza l'installazione del pacchetto in caso di errori.
- **rpm –i <url>**
  Installa il pacchetto a partire dal'URL indicato se non si verificano errori.
- **rpm –ivh <pacchetto-rpm>**
  Installa il pacchetto se non si verificano errori, mostrando qualche informazione e una barra di progressione.
- **rpm –i —replacefiles <pacchetto-rpm>**
  Installa il pacchetto senza verificare se vengono sovrascritti dei file.
- **rpm –i —ignorearch <pacchetto-rpm>**
  Installa il pacchetto senza verificare l'architettura dell'elaboratore.
- **rpm –i –ignoreos <pacchetto-rpm>**
  Installa il pacchetto senza verificare il tipo di sistema operativo.

Comandi per l'upgrade

- **rpm –U <pacchetto-rpm>**
  Aggiorna il pacchetto se non si verificano errori.
- **rpm –Uvh <pacchetto-rpm>**
  Aggiorna il pacchetto se non si verificano errori, mostrando qualche informazione e una barra di progressione.

Comandi per cancellare

- **rpm –e <pacchetto-rpm>**
  Elimina (disinstalla) il pacchetto.

Comandi per la verifica

- **rpm -V <nome-del-pacchetto-installato>**

Verifica che il pacchetto indicato sia installato correttamente.

- **rpm -Vf <file>**
  Verifica il pacchetto contenente il file indicato.
- **rpm -Va**
  Verifica tutti i pacchetti.
- **rpm -Vp <pacchetto-rpm>**
  Verifica la corrispondenza tra il file RPM indicato come argomento e quanto effettivamente installato.

Dipendenze e altri sistemi di gestione

RPM, nell'installazione dei pacchetti, mantiene tutto l'elenco dei programmi e delle librerie necessari per il corretto funzionamento del software. Se qualcuno degli elementi necessari non esiste, o ha una versione troppo vecchia, il gestore del pacchetto non lo installa e notifica un errore di dipendenza. Queste informazioni si chiamano "dependencies".

Le dipendenze possono essere un fattore critico nelle installazioni, perché non permettono di installare un software fino a quando non si risolvono le dipendenze e non sempre il problema si risolve con la ricerca delle versioni più recenti dei file e delle librerie.

In certi casi, per esempio quando si è sicuri che non ci saranno problemi con le dipendenze segnalate da RPM, si può forzare l'installazione con il comando:

```
rpm -i —nodeps <pacchetto-rpm>.
```

Per dare questo comando bisogna essere veramente sicuri di non avere problemi di dipendenza perché il gestore utilizza questo controllo per garantire che il software che si sta installando funzioni correttamente e non crei conflitti con altri applicativi.

Un problema di dipendenze si riscontra, ad esempio, quando si installa un pacchetto con RPM su una distribuzione Linux diversa dalla Red Hat. Il sistema di registrazione di RPM non vede tutti i file che sono installati in realtà, e notifica un gran numero di errori e dipendenze, pur non essendo tutti reali.

Un'ulteriore caratteristica di grande interesse è la possibilità di creare personalmente dei pacchetti software per distribuirli su altre macchine. Per fare ciò si usa il comando:

```
rpm -bb <file-spec>
```

Il file specificato nel parametro deve essere creato dall'utente, e contiene le directory ed i file che saranno inseriti nel pacchetto, oltre alle informazioni importanti per l'installazione e la gestione.

Vediamo brevemente gli altri gestori di pacchetto presenti nelle distribuzioni Linux più diffuse.

La distribuzione Linux/Debian adotta dei file con estensione **.deb** e il comando che gestisce i pacchetti di questo formato si chiama *dpgk*.

Anche la distribuzione Slackware ha un suo tipo di pacchettizzazione del software che si basa sul formato : *tar+gzip*.

Esistono anche programmi di conversione da un formato di pacchetti ad un altro. Sulla rete si trovano applicativi, come ad esempio **alien,** che permette di passare dal pacchetto in formato *deb* a quello in formato *rpm* oppure *tar+gzip*. Purtroppo, le conversioni non sono esenti da errori, anche perché la diversa organizzazione delle distribuzioni Linux limita fortemente l'utilizzo di questo genere di convertitori; quindi bisogna utilizzarle con cautela e solo in caso di necessità.

Conclusioni

I gestori di pacchetti sono degli strumenti molto utili e sicuramente affidabili. Se si installa un applicativo, non si devono cercare i file chissà dove come invece succede con altri sistemi operativi molto più diffusi. Con RPM si sa esattamente cosa si installa, dove lo si installa e quali problemi potrebbe generare.

L'autonomia di scelta dell'RPM è un'altra forma di libertà che su altri sistemi operativi non esiste, se si sceglie di forzare l'installazione in presenza di errori o di dipendenze, il gestore installa lo stesso il pacchetto e lascia che l'amministratore del sistema si assuma la responsabilità delle sue scelte.

Le interfacce X11 (ad esempio GNOME, di Linux contengono anche degli applicativi grafici che rendono RPM ancora più semplice e potente da utilizzare: scelta di pacchetti da installare con modalità da file manager, query sui pachhetti e sulle installazioni tramite form molto intuitive, verifica dei pacchetti con report molto dettagliati su finestre XWindows.

Forse il tempo dei vari Setup ed Uninstaller è veramente finito. Installare un applicativo, da ora in poi, non creerà più dubbi e angosce sulla incolumità del sistema e sull'instabilità degli applicativi precedentemente montati e consolidati.

RPM è uno strumento in più di quelli che faranno dormire sonni tranquilli agli amministratori di sistema.

Marco Gastreghini

Vai a The Bazaar e trovi il più importante evento "fisico" a supporto del movimento "free and open source". Ci sarà la "comunità": i visionari, i progettisti, i programmatori, i creatori delle fonda-

menta tecnologiche, che fronteggeranno, parleranno, accuseranno, blandiranno, ammalieranno le imprese che propongono servizi e prodotti legati al mondo dell'open source. L'inizio di una dimensione economica reale del movimento passa da questo rapporto e questa, per ora, è l'unico "locus" che prevede un confronto tra le due "anime". Sentirete i visionari discutere del futuro della comunità ma, avrete accesso ad una ineguagliabile opportunità per mettere le mani sui prodotti che stanno radicalmente ridisegnando la geografia di questo territorio. È l'evento del mese in cui sarà possibile incontrare Richard Stallman, Ralph Nader, Michael Cowpland (CEO della Corel), Jon "Maddog" Hall, Direttore esecutivo di Linux International e Bob Young CEO della RedHat ma, anche sentire Rob Malda e Jeff Bates di Slashdot, Eric Raymond, Bruce Perens e Chris DiBona. The Bazaar ha 5 track specifiche tra cui scegliere:

a) Business to Business Track;
b) Beginners Track;
c) Administration Track;
d) Developers Track;
e) Open Track.

Registrazione: $595
Informazioni: the Bazaar Headquarters
E-mail: info@thebazaar.org
Tel. (888) 445-8505 -(201) 689-9354

1-3 Novembre *SANS London 99*
The Information Security Conference,
SANS Institute London http://www.sans.org/
5-6 Novembre *II Congresso Hispalinux*,
Universidad Carlos III, Madrid, España
http://www.hispalinux.es
15-19 Novembre *Linux Business Expo presso il COMDEX/Fall '99*, Las Vegas Hilton, Las Vegas, Nevada http://www.comdex.com
16-17 Novembre *OSDF'99 - Open Source Development Forum*, COMDEX/Fall '99 Las Vegas, Nevada, USA http://www.comdex.com/
17-18 Novembre *Linux Expo '99*, London, UK; 1999 (Tel: +44 (0)1256 384000)
16 -18 Novembre *ESC Europe 1999 - Embedded Systems Conference Europe Maastricht Exhibition and Congress Centre (MECC)*, Maastricht, Netherlands.
http://www.esceurope.com/
Non solo una conferenza tecnica di qualità, ma un luogo d'incontro tra acquirenti e produttori di embedded system. L'occasione per sentirsi al centro della prossima frontiera dei sistemi Linux.

Martedì 16 Novembre
09:30 - 17:30 *Tutorial*
Mercoledì 17 Novembre
08:30 - 17:30 *Conference Classes*
10:30 - 11:30 *Keynote Address*
12:00 - 18:30 *Exhibits Open*
17:30 - 18:30 *Opening night*
20:00 - 22:00 *ESCE Party*
Giovedì 18 Novembre
08:30 - 18:00 *Conference Classes*
10:00 - 16:30 *Exhibits Open*

5-9 Dicembre *XML'99 - Graphic Communications Association,* Pennsylvania Convention Center, Philadelphia, Pennsylvania
http://www.gca.org/
12-15 Dicembre *SOPS ACM Symposium on Operating Systems Principles*, Kiawah Island Resort, vicino Charleston SC
http://www.diku.dk/sosp99/
14-16 Dicembre *The Bazaar*, Jacob Javits Center NYC USA http://www.thebazaar.org
(Vedi informazione a lato)

NOV	D	L	M	M	G	V	S	D	L	M	M	G	V	S	D	L	M	M	G	V	S	D	L	M	M	G	V	S	D	L
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

DIC	M	M	G	V	S	D	L	M	M	G	V	S	D	L	M	M
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

FreeBSD ha percorso una lunga strada dal 1993. Dalla sua genesi come ``Unofficial 386BSD Patchkit" ai giorni odierni in cui dimostra di essere divenuto un potente server adatto ad applicazioni industriali, il passo non è stato né breve, né facile. Una vasta comunità di utenti in tutto il mondo sta a testimoniare l'indiscusso successo dell'approccio. Mancava solo FreeBSDCon, e adesso c'è. Qualche dubbio sul futuro della distribuzione e del rapporto con gli altri sistemi operativi open source, grande la sicurezza di giocare ancora un ruolo decisivo nel panorama dei sistemi operativi, Jordan K. Hubbard, presidente e CEO di FreeBSD Inc., ha colto l'occasione dell'apertura di questa prima conference per rivendicare la storia assolutamente unica di FreeBSD, per colorare, forse a tinte un po' più rosee del necessario, lo "stato dell'unione" e tracciarne il futuro prossimo. L'afflusso di pubblico è stato straordinario, tanto da dover raddoppiare le sessioni di tutorial anche nei due giorni successivi alla manifestazione, e il livello tecnico delle relazioni veramente ottimo.

http://www.freebsdcon.org/

...e tremano i colossi

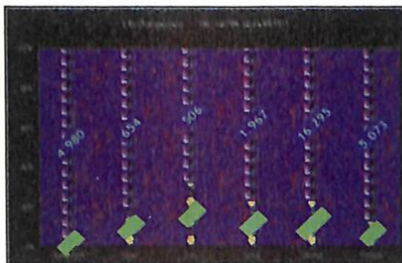



Figura 1: Jim's Public Opinion Research Project. Il grafico è basato sul numero di risultati ottenuti con una ricerca in Alta Vista, per ciascuna delle parole riportata sulle ascisse, seguita direttamente dal nome di uno dei due sistemi operativi Windows o Linux.
http://www.jbum.com/jbum/public_opinion.html

"Linux sta emergendo come un potenziale concorrente di Windows e Unix per svariate applicazioni server". Questa la perentoria affermazione di Dan Kuznetsky, program director per gli ambienti operativi e serverware di International Data Corp. (IDC). Ciò a margine di una ricerca curata lo scorso agosto dalla stessa IDC: su circa 800 utenti residenti in USA e Canada, il 13 per cento utilizzava il sistema operativo Linux. Un'espansione, specificava ancora l'agenzia, destinata ad riflettersi a tutti i livelli in ambito business, grazie anche all'iniziale prudenza con cui quel mondo guarderà a Windows 2000 nei primi 12-18 mesi dal lancio. Ancora, nel 1998 l'utilizzo di Linux quale software per server ha registrato un balzo a dir poco esaltante, in salita del 190 per cento, raggiungendo la cifra di 750.000 unità, per una quota pari al 16 per cento del mercato totale. Nello stesso periodo, la penetrazione di Windows NT è stata del 37,6 per cento (per una fetta di mercato del 38 per cento), mentre quella di Unix si è fermata al 13,7 per cento (con il 19 per cento di share). Infine, da qui al 2003, gli esperti prevedono che la diffusione di Linux avrà un ritmo due volte superiore a quella degli altri sistemi, sempre nell'ambito dei server: il 25 per cento contro il 12 per cento annuale. Proiezioni azzardate? Nient'affatto. Perchè sono i numeri odierni a confermare che Linux ha conquistato qualcosa più di un settore di nicchia, e che insieme agli altri gioielli open source (FreeBSD, Apache, Sendmail) il marketplace globale dell'Information Technology sta seriamente considerando tutto ciò come una valida alternativa a Microsoft. Certo, il business model e la mancanza di una precisa leadership all'interno della community Linux (e open source) sono elementi che fanno ancora storcere il naso a parecchie aziende di Fortune 500. E IBM scopre che il novanta per cento dei responsabili dei network d'informazione della sua clientela di base "è solito far uso di tre o più sistemi operativi." Eppure Mr. Linus Torvalds non ha dubbi: "Nel prossimo futuro Linux effettuerà il sorpasso. Non accadrà in uno o due anni, ma in tre o quattro anni sarà la vera alternativa di mercato per utenti non-tecnici. Diciamo nel 2004, chiunque deciderà di acquistare un computer troverà naturale fermarsi a considerare se vorrà avere installato Linux o Apple o Microsoft." Non basta ancora? Secondo un recente poll ZD/Harris, due terzi degli utenti considera Linux ormai pronto ad affondare il colpo di grazia a Windows. E l'inizio d'autunno conferma che la sua penetrazione ha ormai toccato il 20 per cento del mercato totale, sia per l'ambiente server che per i client. Chiaro: i poll non rivestono alcun valore scientifico. Le opinioni degli esperti neppure. Ma restano questi (e altri) dati a sostegno di un costante, forte incremento generale che non potrà mancare di fruttificare. E, soprattutto, c'è la crescita di coesione che la comunità Linux (e open source) va dimostrando giorno dopo giorno, esempio perfetto della ricca sinergia alla base dello sviluppo del software non proprietario. Un modello aperto e libero che ha ormai varcato con successo perfino le colonne d'Ercole del marketplace. Tremate, colossi, tremate.

Bernardo Parrella

Figura 3: IOS counter (1). Qual'è il sistema operativo più diffuso in rete? L'IOS counter usa "queso" per scoprirlo. Ecco i dati di aprile 1999. Linux saldamente in testa con quasi 40000 unità. Sebbene da gennaio 1999 Microsoft non abbia sostanzialmente perduto terreno bloccandosi al 24% Linux erode un 4% dagli altri Unix: perdono un 1% a testa Sun Solaris, BSD e IRIX e un complessivo di 1% per tutti gli altri, HP, AIX, Sinix, Netware, Digital Unix e SCO. Forza Linux!
http://www.hzo.cubenet.de/ioscount/

Figura 2: The ZDNN/Harris QuickPoll Linux diventerà più importante di Windows? Se lo è chiesto, sulle pagine di ZDNN, la Harris con una indagine. Gli utenti hanno le idee chiare su Linux: "Sì, prima o poi lo farà fuori!" dicono due persone su tre. http://www.zdnn.com/

Figura 4: OS counter (2). Una istantanea del rapporto tra Linux e SO Windows. In Polonia e Germania e Portogallo sono sul 70% i server Linux, in Finlandia e Gran Bretagna circa il 60%, ancora sopra il 50% Olanda e Repubblica Ceca. Italia? Un modesto 39%
http://www.hzo.cubenet.de/ioscount/

La modestia dei rivoluzionari

La comunità Linux (e Open Source) si poggia sulle fondamenta di sempre: entusiasmo collaborativo e comunione d'intenti. Forse ancor più su un elemento centrale, pur se quasi impalpabile: la modestia.

"LinuxWorld, San Josè, California, agosto '99. Nell'introdurre l'attesissimo intervento di Linus Torvalds, Larry Augustin, fondatore e presidente di VA Linux, nonché chairman della conferenza, chiede quanti tra il pubblico abbiano finora dato una concreta mano nella realizzazione del sistema open source. Ad alzarsi in piedi è una buona metà delle diverse migliaia di persone presenti in sala, salutati da fragorosi applausi. Auto-celebrazione?
No, nulla a che fare con simili trabocchetti. Piuttosto, la pubblica riconferma di quella compartecipazione generale che è elemento storico alla base della Linux community. L'età adulta di un impeto "rivoluzionario" che, superata la fase della ribellione giovanile prima, e quella dell'attivazione di un vero e proprio "movimento" poi, oggi va imponendosi come nuova modalità di pensiero, come articolato approccio alle faccende del mondo.
Finora, il modello di sviluppo di Linux ha potuto contare su fondamenta assai solide: la massa critica di entusiasti programmatori a livello globale e l'unificante missione di netta opposizione all'attuale industria del software. Sull'onda di tali affermazioni, ora si assiste alla germinatura di una vasta gamma di pragmatici incentivi che vanno conquistando l'imprenditoria di ogni ordine e grandezza. Software per server Internet (e-mail, Web hosting, server cluster, operatività di network), sistema leggero per una miriade di elettrodomestici d'informazione (dai portatili ai palmtop a svariati dispositivi in arrivo) e oltremodo malleabile per le mille necessità applicative del business moderno.
Superiori alla ventina le aziende che vendono software e servizi Linux-oriented, con altre 35 che producono hardware già appositamente equipaggiato. E sono sicuramente centinaia, se non migliaia, quanti forniscono consulenza e assistenza a vario titolo, sparsi nel mondo, oltre che in Internet. Che dire poi del fulgido ingresso in borsa di Red Hat? Lanciata nel corso del medesimo LinuxWorld di metà estate, la scalata a Wall Street si è aperta alla modica cifra di 14 dollari ad azione, per balzare in pochi giorni a 72, finendo con lo stabilizzarsi, un mese e mezzo dopo, intorno agli 80 dollari. A detta degli esperti, un'esaltante apripista per il prossimo assalto di altre società Linux al grande capitale high-tech. Senza dimenticare, infine, i grossi nomi che continuano ad investire a piene mani in svariati progetti connessi allo sviluppo Open Source: IBM, Compaq, Dell, SGI fino a Fujitsu e Corel. A fronte di tutto ciò (è forse il caso di rammentarlo?), il kernel del sistema operativo rimane gratuitamente disponibile su Internet e altrove, mentre ogni modifica e distribuzione è pur sempre vincolata alla pubblica fruibilità (inclusi ulteriori cambiamenti) garantita dalla licenza open source. Metà sistema operativo, metà movimento sociale, la comunità di sviluppatori Linux va quindi prendendo le misure con le partnership imprenditoriali e il successo commerciale. Ma non lo fa a cuor leggero, priva cioè della coscienza dei rischi insiti in questa fase delicata.
Una questione per tutte: come riuscire a far convivere i relativamente pochi che vanno rapidamente arricchendosi con i programmatori di base, i puristi del movimento? "Questa è una comunità dove Gandhi viene riverito come un eroe," sottolinea Tim O'Reilly, a capo dell'omonimo mini-impero editoriale tutto orientato intorno a Linux. "E' impossibile andar dietro soltanto al denaro senza perdere spazio e credibilità personali." Rincara la dose lo stesso Linus Torvalds: la collaborazione tra la comunità di programmatori, le varie aziende coinvolte e i diversi sponsor alle spalle è pressoché perfetta, fondata com'è sulla comune e radicata convinzione di voler conquistare i mercati e i computer di tutto il mondo. Alla faccia di Microsoft, certamente. Ma ancor più e meglio, grazie alla vitalità delle fondamenta di sempre, entusiasmo collaborativo e unità d'intenti, supportate dalla comune voglia di non strafare e di non vendere l'anima al diavolo. Il tutto fecondato dall'indispensabile qualità che contraddistingue i veri rivoluzionari: la modestia.

Bernardo Parrella "

NOVITÀ IN EDICOLA

Softline CD Mese all'incredibile prezzo di 14.900 lire

Ogni mese in edicola vi aspetta Softline CD Mese, la rivista che per prima in Italia si è occupata del software shareware. Centinaia di titoli vengono selezionati e recensiti per voi scegliendo soltanto il meglio delle novità mondiali. Dai software utili, come i database e le utilità di sistema, ai programmi per divertirsi, dai software scientifici, ai programmi dedicati al mondo di Internet fino ai programmi di grafica più interessanti, ogni mese avrete soltanto l'imbarazzo della scelta. Ed in più un CD interamente dedicato ai giochi 3D. Cosa aspettate? Acquistatelo!

Ogni mese con 2 CD-Rom

EDIZIONI MASTER

Grandi formati
Con licenza d'uso
2519x1679 pixels - 24 bit

400 FOTO D'AUTORE

Liberamente utilizzabili
Altissima risoluzione
Inedite e originali
Di eccezionale qualità
Tematiche
Per sistemi Mac e Win
Include browser per la visualizzazione

2CD L.14.900

In questo numero:
Arte, Still-life, Lavoro, Frutta, Acqua, Fiumi, Costumi, Paesaggi

Photo Bank é un prodotto nuovo, realizzato pensando alle esigenze di tutti coloro che, per lavoro, per hobby o semplicemente per passione, si trovano a lavorare con le immagini e vogliono una fonte nuova, fresca e vitale dalla quale poter attingere a piene mani laddove la semplice creatività non é sufficiente.

Ogni raccolta comprende otto diversi generi fotografici, e ogni genere rappresenta solo il meglio del meglio. Tutte le immagini nascono dall'obiettivo di professionisti affermati ed ognuna di esse racconta una storia che merita di essere parte del vostro lavoro quotidiano.

In Edicola